DISEÑO DE UNA HERRAMIENTA SOFTWARE PARA EL APRENDIZAJE DEL IDIOMA BASADO EN PROCESOS HISTÓRICOS

Proyecto de Sistemas Informáticos

Facultad de Informática

Universidad Complutense de Madrid

Curso 2012/2013

Autores:

Carmen Barroso Puertas

Joaquín Guillamón Martín

Iván Gonzalo Hinojosa

Director:

Ramón González del Campo Rodríguez Barbero

AUTORIZACIÓN

Autorizamos a la Universidad Complutense de Madrid a difundir y utilizar con fines académicos, no comerciales, y mencionando expresamente a sus autores, tanto la propia memoria, como el código, la documentación y o el prototipo desarrollado.

Carmen Barroso Puertas

Joaquín Guillamón Martín

Iván Gonzalo Hinojosa

ÍNDICE

1 CONTENIDO

ÍNDICE DE FIGURAS

2 RESUMEN

Hoy en día existen muchas aplicaciones para aprender idiomas, pero desgraciadamente solo unas pocas merecen la pena y éstas a su vez no son gratuitas. Con éste proyecto, implementado en el lenguaje Java[18] y usando una base de datos en MySQL[19], se intenta dar una pequeña solución a éste problema.

La aplicación está diseñada con dos modos, un modo profesor que permite modificar la base de datos de palabras para así gestionar el contenido que enseñar a sus alumnos, y un modo alumno, donde se le da acceso a una serie de información teórica y ejercicios para su aprendizaje. Además el alumno podrá acceder también a una serie de imágenes y sonidos relacionados con el contenido de las lecciones para así facilitar el aprendizaje de las mismas. El desarrollo de éste proyecto está basado en la metodología de ingeniería del software, por lo que incluimos en la memoria una especificación de requisitos del proyecto.

3 PALABRAS CLAVE

Java, MySQL, base de datos, aplicación, idiomas, sintetizador de voz, mastery learning.

4 ABSTRACT

Nowadays there are a lot of applications for English learning, but mournfully only a few are good enough, and these ones are not free. With this project, implemented in Java language and using a MySQL database, we will try to give a solution for this problem.

The application has been designed with two modes: The teacher mode that allows editing words from the database to manage the content and teach them to the students, and the student mode, where the student user can access to the theory and exercises for his learning. Also, the student can access to flashcards containing images and sounds related to the lesson for an easy learning. The project development was made using the software engineering methodology, so a project requirement specification is included.

5 KEYWORDS

Java, MySQL, database, application, languages, speech synthesizer, mastery learning.

6 INTRODUCCIÓN

6.1 EL VOCABULARIO

El aprendizaje de vocabulario es la herramienta más útil a la hora de introducirnos en un nuevo idioma. Los profesores reconocen la importancia del papel que juega el repertorio léxico a la hora de dar los primeros pasos en una lengua ajena, pero nunca ha habido consenso en cómo hacer uso eficiente de la introducción de nuevos términos entre los alumnos.

A lo largo de los últimos años han aparecido numerosos estudios sobre este apartado de la competencia en comunicación lingüística, y se han obtenido abundantes conclusiones de los datos empíricos obtenidos para mejorar en le receptividad de nuevas palabras.

6.2 IMPORTANCIA DEL VOCABULARIO

El vocabulario desempeña un rol esencial en la vida humana y en sus posibilidades. Un vocabulario extenso y rico suele estar asociado a una persona bien formada, y cuanto más amplío es el repertorio de la persona, suele estar relacionado con una buena comprensión lectora y rendimiento académico.

Entre los muchos estudios y conclusiones contenidas que lo avalan, hay datos obtenidos esclarecedores relacionados con el vocabulario, grupos de habilidad y procedencia socioeconómica de los estudiantes (M.F. Graves[4], 2009; Beck[1], McKeown y Kucan, 2009):

- El conocimiento del vocabulario en preescolar y primer curso es un claro predictor de la comprensión lectora tanto en primaria como en secundaria.
- La dificultad del vocabulario influye en la legibilidad de los textos.
- El vocabulario que usamos influye poderosamente en lo que los demás piensan sobre nuestra competencia.
- La enseñanza del vocabulario mejora la comprensión lectora.
- Un vocabulario limitado puede explicar el fracaso escolar de muchos alumnos.
- Los alumnos de 1º de primaria de nivel socioeconómico alto conocían el doble de vocablos que los de nivel bajo.
- Los alumnos de final de la enseñanza secundaria con alto rendimiento conocían cuatro veces más palabras que sus compañeros de bajo rendimiento.
- Los alumnos de 3º primaria con alto rendimiento tenían el mismo nivel de vocabulario que los de rendimiento más bajo al final de la enseñanza secundaria.

Para poder salvar esas diferencias, que en la mayoría de los casos se vuelven permanentes, la docencia debería dar prioridad en la adquisición de vocabulario de una manera más efectiva, ya que este se prueba imprescindible en el aprendizaje de las cuatro habilidades lingüísticas básicas: hablar, escuchar, leer y escribir.

En la enseñanza del vocabulario, hay que distinguir los cuatro diferentes grupos de palabras del que se compone el vocabulario de cada estudiante:

- Las palabras que se comprenden cuando se escuchan
- Las palabras que se pueden leer
- Las palabras que puede usar en expresión oral
- Las palabras que usa en expresión escrita

El vocabulario en el lenguaje escrito es mucho más extenso y variado que el usado en el lenguaje oral, y su riqueza léxica es también mucho mayor. Es por ello que la lectura es la principal y más eficiente fuente a la hora de añadir nuevas palabras a

nuestro acervo, por delante de la expresión oral, que está en constante renovación a lo largo de nuestra vida ampliándolo y perfeccionándolo día a día.

Puede adoptar 2 formas:

- Vocabulario receptivo: El que reconocemos y comprendemos. Es más extenso, y es importante por el hecho que el añadir nuevas palabras ya pertenecen a nuestro vocabulario oral, la asociación facilitará el aprendizaje.
- Vocabulario productivo: El utilizado escrito y hablado. Es menos extenso que el receptivo, y lo forman las palabras más comunes y cotidianas, mientras que el receptivo posee las más raras y complejas.

6.2.1 HECHOS CIENTÍFICO NOTABLES:

- Año 2000. Se publica uno de los trabajos más influyentes sobre la enseñanza y aprendizaje de la lectura: Report of the National Reading Panel : Teaching Children to Read[5] (National Institute of Child Health and Human Development, NICHD, 2000) que ha tenido un gran impacto en la legislación y en la política educativas. Surgió de un mandato del Congreso de los Estados Unidos con el propósito de evaluar la investigación científica realizada sobre la lectura (enfoques y métodos) y sus implicaciones en la enseñanza/aprendizaje de la misma. Las conclusiones del Panel: cinco componentes son esenciales para la enseñanza de la lectura: conciencia fonémica, correspondencia fonema-grafema, vocabulario, fluidez y comprensión. Otorga, pues, al vocabulario un papel central que tendrá importantes consecuencias para el papel que ha de ocupar en el currículo.
- Año 2003. En el caso de EE.UU., el Departamento de Educación, consciente de la necesidad de profundizar en el aprendizaje y la enseñanza del vocabulario, patrocinó una conferencia (Focus on Vocabulary) que tuvo lugar en Dallas (Texas) a primeros de octubre de 2003. Las aportaciones de los expertos asistentes dieron lugar a la publicación de un libro imprescindible en el campo de la enseñanza y aprendizaje del vocabulario: Teaching and Learning Vocabulary. Bringing Research to Practice, editado por Elfrieda H. Hiebert y

Michael L. Kamil[3] en Lawrence Erlbaum Associates en el año 2005.

6.3 CÓMO SE APRENDE EL VOCABULARIO

El aprendizaje de vocabulario desde primaria hasta la adolescencia avanzada tiene un ritmo trepidante, tanto que hasta que uno no pone negro sobre blanco los número no puede ser consciente de ello: Se estima (Cunningham[8], 2005) que los niños manejan en torno a los 8.000 vocablos a la temprana edad de 6 años, mientras que al terminar el bachillerato superan holgadamente las 40.000 palabras. Con estos datos en mano, un estudiante medio aprende durante el sistema educativo unas 32.000 palabras durante 12 años, lo que nos da unos 2666 palabras aprendidas por año, o lo que es lo mismo, 7 palabras nuevas por día, durante todos los días. Sin embargo, un análisis detallado del contenido en vocabulario de cada curso, nos indica que los vocablos nuevos introducidos en la enseñanza especifica de vocabulario no son más que unos centenares, por lo que tenemos que explicar de dónde provienen las restantes palabras que se aprenden de manera indirecta o no intencionada.

La solución más aceptada entre los expertos dice que la mayor parte del aprendizaje del vocabulario ocurre en los llamados "encuentros incidentales" con el lenguaje. Estos encuentros se basan en dos partes esenciales: La exposición al lenguaje oral del alumno, y la exposición al lenguaje escrito del alumno. Podemos deducir de esto que la calidad y acervo del léxico de las personas será directamente proporcional a la riqueza del contexto comunicativo en el que se hayan desarrollado, y de la lectura y a la que haya tenido acceso. Aun así, esto no aparta en ningún momento de la valía de la enseñanza sistemática del vocabulario que ha de ejercerse en la escuela como una de las principales causas para el desarrollo léxico de las personas.

Norbert Schmitt[9] (2000), un reconocido autor en el campo de la enseñanza de vocabulario, afirma que hay 2 procesos fundamentales en la adquisición de vocabulario: Lo que denomina aprendizaje explícito por medio del estudio de palabras, y el denominado aprendizaje incidental, a través de la exposición del alumno al uso del lenguaje.

Por todo esto, podríamos clasificar en tres grandes medios las vías por las cuales se aprende el vocabulario según Schmitt:

- El lenguaje oral en el que se desarrolla en individuo.
- El lenguaje escrito, principalmente la lectura, con el que el individuo se relaciona.
- La enseñanza sistemática y directa del vocabulario.

Por todo esto, se aconsejan seguir unas pautas a nivel de aula en el ambiento escolar para estimular el desarrollo de vocabulario de los alumnos:

- Crear en el ámbito del aula un clima rico y motivador sobre la importancia de la comunicación oral, en el que el profesorado aparezca como modelo de aprendizaje ilusionado de nuevos vocablos (Blachowicz y Fisher[2], 2006) y como realizador de una expresión oral que sirva de ejemplo a sus alumnos.

- Promover la lectura en sus variadas formas y de acuerdo con el nivel escolar de que se trate: lectura oral dirigida tanto al grupo de los alumnos de la clase como a algunos de ellos en particular; lectura independiente por parte del alumnado, cuya contribución al desarrollo del vocabulario es definitiva. Sobre la trascendencia de la lectura independiente, sirven de ejemplo estos datos: un alumno que dedica 65 minutos cada día a la lectura independiente puede leer 4.358.000 vocablos al cabo de un año; cuando el tiempo diario dedicado a la lectura es de, aproximadamente, 10 minutos, las palabras leídas al año serán 622.000 (Anderson, Wilson y Fielding[7], 1988).

- Sobre la enseñanza directa del vocabulario, digamos por ahora que se han producido importantes cambios de orientación en los últimos años. No son recomendables las listas de vocablos a aprender. Hay que fomentar el aprendizaje generativo de los vocablos en los contextos textuales en que aparecen.

6.4 APRENDIZAJE EFECTIVO DEL VOCABULARIO

La dicotomía sobre la importancia primaria entre la gramática y la pronunciación en el aprendizaje es una cuestión residente en la enseñanza por mucho tiempo. Aunque

ambas tienen su función e importancia necesaria, es evidente que con un mínimo de pronunciación, entender y expresarse a mejor nivel en un idioma vendrá dado por el vocabulario de cada uno, siempre entendiendo el vocabulario como reconocer, entender y reproducir una palabra.

Indicador de ello es hacer un ejemplo simple entre una frase gramaticalmente incorrecta y otra con vocabulario incorrecto: En castellano, cualquiera persona entendería a un turista si nos preguntara "¿Estar dónde iglesia?", pero si nos dijera "¿Por favor, dónde está la church?" solo le entenderíamos si supiéramos inglés, con lo que a la primera frase gramaticalmente incorrecta podríamos dar respuesta de inmediato, mientras que en la segunda deberíamos cubrir el hueco dejado por la falta de vocabulario con más explicaciones. Es cierto que en frases más largas una ligera ausencia de vocabulario se puede solventar intuyendo la palabra por el contexto, pero si falta más de una quinta parte de las palabras se hace prácticamente imposible hacerse entender.

Es por todo esto que la memorización de vocabulario es un factor esencial en el aprendizaje de una lengua, y aunque es cierto que cada persona tiene sus habilidades y preferencias respecto a los métodos, los estudios[12] [14] [15] han racionalizado y clasificado todos ellos dependiendo del tipo de persona, memoria y eficiencia efectiva de cada uno de ellos.

6.4.1 APRENDIZAJE EMOCIONAL

Los humanos somos animales emocionales: Es sabido que los recuerdos a largo plazo más permanentes son todos aquellos que han tenido mayor mella en nuestras emociones que aquellos sucesos que no nos han afectado a nivel personal tanto. Las lecciones dadas por profesores divertidos y que alientan a los alumnos, como las que son impartidas por profesores más odiados por su rigidez y complicación de la materia, resultan más fáciles de recordar que las clases impartidas con profesores más aburridos o que no generan ningún tipo de empatía positiva o negativa en los receptores, y esto es porque tanto los sentimientos positivos o negativos influyen en la emoción del recuerdo a la hora que la memoria guarde mejor la información impartida.

Además de la memoria a largo plazo antes mencionada, la memoria también consta de una parte dedicada a la retentiva a corto plazo. Está, apenas unos segundos de visualizar un término nuevo, queda almacenada sin ningún problema y podemos recordarla sin dificultades, pero dependiendo de cada persona, a partir de un paso de tiempo relativamente corto, como 2 o 10 minutos, hace que sea complejo recordarla. Haciendo una analogía con los computadores, la memoria a corto plazo actúa como una memoria RAM, que almacena temporalmente hasta un límite temporal y de cantidad la información que captamos, y la de largo plazo sería el disco duro en el que la memoria la tenemos bien recaudada y ordenada a largo plazo, aparentemente sin límites de tiempo, pues estos recuerdos pueden durar meses, años o para toda la vida.

La clave está en cómo gestionar el flujo de vocabulario hacia nuestro disco duro en vez de a nuestra memoria RAM para mejorar el aprendizaje. A veces nos encontramos que guardamos información irrelevante o de importancia nula en nuestra memoria a largo plazo mientras que con la información que querríamos no lo conseguimos, y se debe a que aunque el concepto recordado nos resulte irrelevante en sí, ha activado un mecanismo emocional que ha conllevado a que lo recordemos con más facilidad. Ejemplo de ellos pueden ser los jingles de anuncios publicitarios de televisión que se nos quedan dando vueltas en la cabeza, cuando recordamos una palabra por una foto o imagen que vimos al leerla, o cuando la palabra lleva asociado un momento cómico o especial cuando la descubrimos. En todos esos casos, la memorización fue más efectiva debido a la asociación producida en el momento de recepción del dato (visual, auditiva, táctil,…), y la marca emocional que ello conllevo y que propicio su memorización efectiva.

Aún con todo y ello, esto solo no serviría, puesto que el desuso y la capacidad el cerebro no es infinita para albergar todos los datos emocionales que recibimos, si no podríamos llegar a saturarnos en el día a día con demasiada información acumulada y tener trastornos complejos, como el síndrome hipertiméstico o el síndrome del sabio, que quienes lo padecen gozan de una claridad y profundidad de memoria inacabable, pero que requieren de tratamiento psicológico y médico en la mayoría de los casos por cefaleas y migrañas. Es por ello que cada tanda de aprendizaje nuevo conlleva una pérdida irremediable de más del 50% del vocabulario, y se necesita de más métodos para

ayudar al aprendizaje efectivo de estas tandas.

6.4.2 APRENDIZAJE RELACIONAL

Entre los métodos que ayudan a fija los nuevos términos se encuentra la decoración: Consiste en asociar un término con todo lo relacionado a nivel sensorial que nos venga en mente a la hora de hacerlo, ya sean diagramas, dibujos, sonidos, palabras parecidas, juegos de palabras, narraciones, pronunciarla en voz alta o semejantes. De esta forma, facilitará el paso de las palabras de la memoria a corto plazo en la memoria de largo plazo, ya que cuantas más asociaciones o referencias añadamos en la palabra, más importancia le estaremos dando al término, puesto que el esfuerzo de asociación facilita nuevas vías de acceso a la palabra en nuestra mente.

6.4.3 APRENDIZAJE POR REPETICIÓN

Para recordar algo con éxito, debemos pasar un largo tiempo sin pensar en ella para que la memorización haya sido efectiva, es decir, espaciar las repeticiones. Pensar en algo constantemente por un periodo corto de tiempo no es recordarlo, puesto que lo único que lograremos, como mucho, es alargar su estancia en la memoria a corto plazo. Suelen funcionar de manera efectiva los intervalos más espaciados, con un primer intento 3 minutos después de la memorización, un segundo intento alrededor de la hora posterior, y un tercer intento final de comprobación una semana más tarde de ello. Es por eso que los repasos periódicos en los métodos de estudios ayudan mejor en la eficacia de aprendizaje, dado que esa pauta repetida ayuda a fijar en memoria en caso de que la asociación haya fallado o no hubiera quedado del todo fijada.

Es también digno de recalcar que bajo ningún concepto han de forzarse estás sesiones de recuerdo, o hacerlo bajo presión o en ambiente desfavorables. La razón más importante por la que los niños tienen más facilidad e aprendizaje que los adultos es la motivación a la hora de hacerlo: mientras que los niños aprenden casi sin querer, entre juegos y divirtiéndose, lo adultos tienden a intentan aprender “porque sí”, siendo esta razón un obstáculo evidente en la efectividad del aprendizaje. Es por esto de la tendencia actual de la gamificación[16] en los entornos de aprendizaje, ya que con

mecánicas de juego, diversión y factores sociales, ayudan en la motivación extra necesaria para mejorar la calidad de la docencia y mejorar los resultado[16]

Según un estudio realizado por la Universidad británica de Sussex[13], repetir una y otra vez el mismo cuento en la infancia de los niños favorece su aprendizaje de vocabulario, y se acelera la adquisición de los nuevos términos desconocidos. La psicóloga Jessica Horst[10] dirigió el estudio, y sus resultados son consecuencia de un experimento en el que se expuso a dos grupos de niños de 3 años al aprendizaje de dos palabras nuevas. Cada una de las nuevas palabras, eran palabras inventadas para designar un objeto desconocido. Durante una semana, uno de los grupos de niños escuchó tres historias totalmente diferentes con estas palabras y el otro grupo sólo una historia repetidas veces con las mismas palabras nuevas. Tras una semana, se observó que los niños que habían escuchado el mismo cuento una y otra vez recordaban mucho mejor las nuevas palabras que los niños que habían escuchado tres cuentos diferentes. Esto se debe a que en una primera lectura los niños comprenden la historia, en la segunda perciben los detalles y la descripción y a cada nueva lectura van ahondando en la profundización del entendimiento. Previamente, la psicóloga ya había comprobado en otro estudio similar que los niños que ven el mismo programa de televisión o película una y otra vez son los que obtienen mejores resultados en los exámenes de comprensión.

6.4.4 APRENDIZAJE POR CAPTACIÓN VISUAL

Como antes hemos mencionado, la decoración de las palabras ayuda en el proceso de aprendizaje, tanto por su faceta relacional como la capacidad emocional en la relación de lo que decoremos las palabras. En esta faceta se basa la memorización por escenas, una variante de la memorización con apoyo visual, también utilizado en el método de tarjetas o flashcards. Se trata de aprender utilizando tarjetas o cartas con una imagen y una palabra a recordar, por lo que la asociación viene dada de antemano, y funciona como cebo visual: El lector instintivamente relacionara la palabra con la imagen, y hará una asociación de ellas con lo que percibe a primera vista, dándole un significado. Una vez visto, el cerebro puede hacer uso de la imagen como enlace a la palabra, siendo más fácil de recordar.

El uso de los colores juega un papel muy importante en este aprendizaje relacional. Los colores son uno de los efectos visuales más potentes que puede haber a nivel visual, y se puede hacer uso de ello en el aprendizaje de vocabulario creando lo denominado como mapas mentales, asociando la información como un flujo de líneas de colores que hacen las veces de abstracción de datos y clasificación. Se puede hacer uso de colores tanto en la relación de imágenes con palabras, como para la clasificación de géneros, números, verbos y adjetivos en idiomas en los que el análisis léxico sea complejo, y creando un acceso rápido a la solución en caso de duda.

6.4.5 APRENDIZAJE POR CORRECCIÓN

El método ensayo-error, o tratar de forzar respuestas erróneas, es otra alternativa a la hora de reforzar la memorización. El cerebro tiende a relajarse ante estímulos conocidos o información ya procesada, por lo que forzar una llamada de atención fallando en algo no esperado, o tratar de aprender palabras de una lección no conocida, sirve de acicate para forzar las relaciones de información parecida ya procesada en el pasado, tratando de encontrar similitudes léxicas o parecidas. En caso de fallar, no todo es negativo, puesto que las correcciones ante problemas en las que el cerebro ha estado expuesto durante más tiempo sirven como nuevo vínculo de relación y facilitan el aprendizaje de las palabras erradas en la memoria a largo plazo.

6.4.6 APRENDIZAJE POR CAPTACIÓN AUDITIVA

En cuanto a la asociación auditiva se refiere, es muy corriente la práctica hoy en día de visionado de series y películas con subtítulos en su idioma original. Aunque es cierto que el apoyo visual para el entendimiento de vocablos de pronunciación confusa o de palabras desconocidas en evidente, tiende a favorecerse la memorización visual de la palabra en contra de la captación por el sonido, lo que a la larga desfavorece en la asociación relacional del término en memoria. Es por ello que se recomienda la exposición aislada a la palabra en sus diferentes estímulos para fortalecer el nexo, y se sugiere escuchar las palabras sin los subtítulos o con los ojos cerrados para forzar la asociación en memoria de sonido y termino, para no depender de la asociación palabra termino.

6.5 METODOLOGÍAS DE ENSEÑANZA

6.5.1 APRENDIZAJE ESTANDARIZADO

Los grados son la medida de estandarizar la cantidad y calidad de la educación impartida. Se fijó la cantidad de tiempo que se le dedica a la educación en años, y se fijó el grado a un año de instrucción. Esta normalización obligó al estudiante a un molde donde se ignoraban las aptitudes y habilidades personales de cada individuo, y como alumno era diferente en cuanto a capacidad de aprendizaje en ese mismo periodo, se inventó el sistema de calificaciones de la A a la F o de 0 a 10 en función del éxito de cada alumno en el aprendizaje. Con estas calificaciones, se conseguía agrupar a alumnos de habilidades y edades similares, cambiando grupos año a año en función de las capacidades y haciendo repetir a los menos "eficientes".

Este sistema falla debido a que limita al estudiante que sobresale de la norma pues restringe lo que puede aprender en el tiempo disponible, y frustra al estudiante por debajo de la norma pues se ve fuera de lugar debido a sus habilidades. No solo eso, las habilidades de cada individuo también varían por materias, dado que un estudiante rezagado en matemáticas podría ser brillante en arte y estar condicionando su aprendizaje en sus materias predilectas además de en las que carece. Por ello, los grados fuerzan a ser todos iguales cuando no lo son, provocando frustración en la mayoría de los estudiantes.

Las calificaciones se crearon para medir el nivel de progreso de cada estudiante respecto al estudiante "estándar" que se tomaba como referente. Representan el porcentaje del material establecido que se aprende en el tiempo prefijado de un año. Generalmente si la calificación es superior al 70% se considera que estudiante es apto y avanza al próximo grado, tomando el 30% restante del material no aprendido como "error del sistema". En otros sistemas se usan las letras del alfabeto para medir el grado de aprendizaje, siendo: A= 90%-100%, B=80%-89%, C= 70%-79%, D=60%-69%, y F=0%-59%. Este sistema hace que el estudiante deficiente permanezca deficiente en todos los grados dado que no ofrece alternativas de apoyo y remedio a los rezagados, y los estudiantes suelen tender a rondar el mismo nivel de calificaciones año a año segregando a los estudiantes según su nivel respecto al grado prefijado.

Este sistema provoca una baja autoestima y frustración a los estudiantes menos eficientes, y provoca que quieran abandonar la enseñanza no sin antes provocar problemas de indisciplina en las aulas y entorpecer el trabajo del maestro con los demás alumnos. La justificación de ello siempre suele recaer en los alumnos y su indisciplina, cuando la verdad es que el problema también recae en el sistema de educación.

6.5.2 APRENDIZAJE INDIVIDUALIZADO: MASTERY LEARNING

Hoy en día, la enseñanza de idiomas y carreras posee un sistema de aprendizaje en el cual el estudiante promedio consigue aprobar aun ignorando un 40% de la materia a aprender, en muchos casos con graves deficiencias de nociones teóricas evidentes. Se califica a los estudiantes en grupos basándose en su nivel de destreza, y provoca que los menos eficientes se sientan apartados, y en muchos casos hacen que se vuelvan indisciplinados y lleguen a odiar la educación y el aprendizaje. Esto es debido por homogenizar a todos los estudiantes para que todos y cada uno de ellos aprenda el mismo número de conceptos en el mismo periodo, y choca con la realidad de un grupo heterogéneo de estudiantes causando que muchos de ellos fracasen.

Para corregirlo se propuso el sistema mastery learning[22], que en vez de obligar a todos los estudiantes a adaptarse al mismo temario con el mismo tiempo de aprendizaje, se adaptará personalizádamente al tiempo y habilidades de cada alumno y sacar lo mejor de ellos. Es un sistema flexible que ayuda a mejorar el conocimiento de cada alumno, refuerza su autoestima y provoca en los individuos una actitud de aprender positiva al inducir un pensamiento de refuerzo de positivo hacia el aprendizaje. En uso desde hace más de 90 años en mucho colegios norteamericanos, ha dejado de considerarse experimental y hay diversos estudios que resaltan sus beneficios respecto al sistema tradicional (*Block (1971), Bloom (1968)*.

Debido a que la individualización de la enseñanza en la época previa a los ordenadores resultaba tediosa y complicada de administrar para los profesores, en comparación con la enseñanza más globalizada y estandarizada que se implantó tras la expansión de la enseñanza pública tras la revolución francesa, ha tardado en despegar y triunfar. Aunque empezaron en 1960 en usar asistencia por computadora usándolas como libros flexibles, y más tarde con bases de datos mantenidas con el ordenador, el uso hoy

en día de sistemas con inteligencia artificial que se ocupan de la tarea administrativa ha terminado por popularizar esta enseñanza.

La enseñanza individualizada no ajusta el individuo al tiempo, sino al contrario. Todos los estudiantes aprenden el material seleccionado de una manera casi total, con menos "error" que en el aprendizaje estandarizado, y la mayoría de ellos acaban siendo de "A", siendo el único factor que cambia el tiempo que cada estudiante tarda en llegar a ese nivel de aprendizaje. Además la cantidad de material para aprender y evaluar se reduce a lo que se puede aprender en periodos de tiempo más breves y no en años lectivos, por lo que se divide en 12 grados, dividiendo cada grado en aproximadamente 150 unidades de aprendizaje. Esto logra reducir la penalización en casa de aprendizaje fallido del alumno al periodo corto de tiempo de la unidad de aprendizaje, y no a los meses o curso como en los sistemas estandarizados.

No solo se dividía por unidades de tiempo diferentes, si no que este sistema también tiene en cuenta las diferentes habilidades y aptitudes de los alumnos en las diferentes materias, adecuando el ritmo de cada alumno en cada materia de maneras más eficiente. De esta manera no se limita y fuerza al alumno, se induce el aprendizaje en el estudiante, manteniendo su motivación alta y haciéndolo destacar en sus áreas más hábiles.

En el caso del aprendizaje de idioma, el mastery learning significaría dividir el lenguaje en diferentes grados por nivel de capacidad en el idioma, habiendo en cada grado múltiples unidades de aprendizaje, siendo cada unidad de aprendizaje un bloque concreto del idioma, ya sean bloques de gramática, sintaxis, vocabulario u ortografía. En nuestra aplicación, el grado es cada nivel de Inglés adquirido en grado UCM (0-5, equivalente a A1-C2 en EOI), y cada lección de cada grado es una unidad o bloque gramatical de vocabulario. El grado de proficiencia por unidad está fijado por la cantidad de veces sin error se acierta cada palabra o frase preguntada, y cuando todas ellas se aprenden se le permite avanzar al siguiente bloque. En el sistema de mastery learning se les provee a los estudiantes información específica acerca del progreso, por lo que usaremos barras de progreso por palabra, lección y nivel para informárselo. Esta información ayuda al estudiante a identificar lo que ha aprendido y recordarle en que

está fallando más y reforzárselo volviéndoselo a preguntar.

6.5.2.1 NORMAS ESENCIALES DEL MASTERY LEARNING:

- El nivel de mastery es definido como 90% en las pruebas de unidad. Este nivel de mastery se puede variar a 100% en algunas unidades simples o muy elementales.
- Las unidades serán clasificadas según la taxonomía del conocimiento de Bloom, para indicar la complejidad de la unidad. La unidades clasificadas en los últimos tres niveles de la taxonomía servirán los objetivos generales de la educación.
- En cada materia, el currículo de primero de básica hasta el último año, en el sistema actual, será dividido en unidades en secuencias que irán creciendo en amplitud de contenido o dificultad de aplicación.
- Esta secuencia de instrucción será dada a conocer al estudiante, sus padres o tutores, y a cualquier persona que lo desee. Esto permite a los agentes involucrados conocer con precisión el mapa de lo que se pretende enseñar.
- Una unidad debe cubrir material que se pueda aprender en aproximadamente 10 horas de instrucción para el estudiante promedio.
- Los exámenes de evaluación de la unidad se pueden tomar en cualquier momento que el estudiante lo desee.
- Completar todas las unidades es completar la educación básica y media en esa materia. Independientemente del tiempo empleado para completarlas
- Las unidades en que el estudiante no obtenga mastery, serán repetidas por este, antes de avanzar a la próxima unidad.
- Se recomienda que al repetir una unidad se use un estilo de instrucción diferente al que se empleó la vez en que el estudiante fracasó la unidad.

Un análisis de los estilos de aprender del estudiante puede ayudar a facilitar aprendizajes futuros. Para facilitar la clasificación de estilos de enseñanza se puede utilizar la taxonomía de Gagné.

6.5.2.2 HECHOS CIENTÍFICO NOTABLES:

- Guskey y Gates (1986) realizaron un meta-análisis de 27 estudios cubriendo cinco áreas: proficiencia estudiantil, retención, variables relacionadas con el tiempo, y variables relacionadas con el maestro.

- Guskey y Pigott (1988) hicieron un meta-análisis en un intento de responder varias preguntas acerca de mastery en un contexto grupal como la efectividad del aprendizaje mastery, que niveles de aprendizaje eran los más influenciados por el mastery, la efectividad en función de la materia o la influencia de la duración del temario, edad o grado en los resultados. Los autores comenzaron analizando 1000 artículos de investigación y redujeron el número a 46. Los efectos positivos de mastery learning fueron hallados en todos los niveles educativos, y en estudiantes jóvenes de las escuelas elementales fueron mayores que en los estudiantes de secundaria y universitarios.

- Kulik, Kulik y Bangert-Downs (1990) condujeron un meta-análisis de 108 evaluaciones de programas de mastery learning. Las medidas evaluadas fueron logros en los exámenes al fin de la instrucción, actitud hacia la enseñanza, actitud hacia el contenido, y proporción que completó el curso. Los logros en los exámenes al fin del curso fueron positivos.

- Clark, Guskey, y Benniga (1983) examinaron los efectos de mastery learning en logros y motivación. El estudio examino un grupo en mastery learning y otro grupo que usó el método tradicional de instrucción de recitación. La variable principal para este estudio fue el efecto de la motivación en los logros estudiantiles. Estos autores encontraron que el grupo en mastery learning obtuvo mayores logros en conocimientos, tuvo

menos ausencias, y estaban más motivados a aprender el material del curso.

- Okay (1974, 1977) examinó los materiales necesarios para enseñar mastery learning, las actitudes de los maestros y la de los estudiantes, y los logros estudiantiles. En todas las variables los efectos fueron positivos. Se encontró que los maestros incorporaban nuevas estrategias en su enseñanza, y esas estrategias influyeron positivamente en ellos mismos y en las actitudes hacia la enseñanza de los alumnos.

- Patterson (1993) describió los esfuerzos de reestructuración en una escuela secundaria de Chicago basado en demandas por estándares más altos, y mayores logros estudiantiles. Esta escuela abandonó las normas viejas y adopto estándares de mastery learning. Resultados del cambio es que más estudiantes están teniendo éxito, debido a expectativas mayores y horario que se acomoda a sus necesidades. Los maestros son interrumpidos menos, y pueden asegurarse que los estudiantes tienen un buen entendimiento del material cubierto en clase. Los estudiantes están obteniendo calificaciones más altas, y más estudiantes están prosiguiendo hacia la universidad.

- El Chicago Mastery Learning Reading Program (CMLR, Programa de Chicago para Aprender a Leer con Mastery Learning) se usa en muchas escuelas. La Junta Educativa de Chicago desarrollo este programa de instrucción para estandarizar mastery learning como instrumento educativo en la enseñanza de lectura en las escuelas de la ciudad. El programa se usa desde parvulario hasta el octavo grado. De estos programas exitosos emergen estos conceptos 1) Mastery Learning es un modelo que tiene éxito con un amplio espectro de estudiantes. 2) Mastery Learning reduce la diferencia académica entre los estudiantes lentos y rápidos, sin disminuir la velocidad de aprendizaje de los estudiantes rápidos. 3) Las habilidades y conceptos aprendidos se asimilan y son usados en otras áreas de estudio. Junto con las mejoras académicas las actitudes y la auto-imagen de los estudiantes también han mejorado.

- La Comisión Para Obtener las Habilidades Necesaria (SCANS en inglés, 1991) del Departamento de Trabajo (Estados Unidos) en un reporte bosqueja los requisitos para el éxito en el ambiente de tecnología cambiante de hoy día. El objetivo de SCANS es crear los fundamentos básicos para el éxito en el trabajo. Estos fundamentos son: habilidades básicas, habilidad de pensar, y buenas cualidades personales. La investigación e implementación de programas de mastery learning muestran que este produce mejoras en cada uno de esos fundamentos. Usando los programas de mastery learning en los fundamentos básicos se pueden lograr los resultados deseados con la gran mayoría de la población estudiantil.

- Los sistemas escolares deben reconocer que los métodos tradicionales de enseñar y aprender no satisfacen las necesidades de muchos estudiantes. Mastery learning es una alternativa a esos fracasados sistemas tradicionales. Robinson (1992) declara que el cambio de los métodos tradicionales a métodos más exitosos va a requerir una reestructuración de cómo las escuelas están organizadas, cómo los maestros son preparados, y cómo pueden ser exitosos. Los sistemas escolares tienen la tarea de definir cómo se va a obtener el éxito en el siglo XXI, implementar esos programas, evaluar los resultados, y decidir cómo satisfacer cada necesidad que surja.

6.6 LA COMPUTADORA COMO ASISTENTE EN EL APRENDIZAJE

Aunque el mastery learning se puede implementar usando lo que se ha llamado enseñanza multigrado y un sistema de fichas que complementan los libros de texto, en este proyecto nos basaremos en que la enseñanza se individualice y personalice a las capacidades de los alumnos con la ayuda de una aplicación para ordenador que sirva al estudiante como libro de texto de vocabulario e idiomas, con una base de datos con su progreso. El papel de maestro como pedagogo directo disminuye, puesto que solo configurará la aplicación con las lecciones de vocabulario y gramática necesarias, pero

aumentará su capacidad como administrador del temario y progreso de los alumnos, además de motivar su aprendizaje de manera autodidacta.

Las computadoras han sido usadas en los últimos 30 años como libros inteligentes que facilitan la instrucción y la individualización de la enseñanza. Sus usos y aplicaciones han sido estudiados con diferentes modelos y los resultados han sido prometedores. Aun así, dado que las computadoras evolucionan constantemente, cada día surgen nuevas ideas de cómo usar su cada vez más avanzada capacidad de cálculo e inteligencia artificial para diseña modelos de enseñanza cada vez más personalizados y eficientes. La última tendencia es a unir los algoritmos de inteligencia artificial a los distintos modelos educativos para imitar la habilidad humana de hacer decisiones.

En un artículo de Kyaw Soe, Ph.D, Stan Koki, and Juvenna M. Chang, Ed.D, cito: "El veredicto para el uso de computadoras en educación ya se ha dado. El Nacional Center for Education Statistics (NCES, 1999) declaró: *Las computadoras se han convertido en una herramienta esencial para nuestra sociedad. La exposición temprana a las computadoras ayudará a obtener la alfabetización en computadoras que es esencial para el éxito en el trabajo. Tener computadoras en la casa y en la escuela permite a los estudiantes recoger información, manipular data, producir resultados eficientemente en nuevas maneras. Examinar hasta donde los estudiantes tienen acceso a las computadores puede ser un indicador de cuan bien preparados están los estudiantes para entrar a los trabajos que cada día son más tecnológicos."*

7 ESTADO DEL ARTE

7.1 APLICACIONES ONLINE DE APRENDIZAJE

Hoy en día, Internet es la primera fuente a la hora de proporcionar material, herramientas y ayuda a la hora de aprender un nuevo idioma. Con la socialización de contenidos, hay una ola ingente de webs de aprendizaje social, donde el usuario ayuda a corregir ejercicios de otros usuarios a cambio de que otros hagan lo mismo que él. Además, el repertorio de sonidos de palabras en lenguaje nativo es una gran faceta a favor. Muchas de las web sociales disponen de aplicaciones o herramientas para el

aprendizaje de vocabulario desde móvil, pero la herramienta principal es mediante el idioma social online.

La segunda de las herramientas más extendidas son las webs de aprendizaje mediante flashcards o modificaciones de la idea de tarjetas visuales de vocabulario/frases con contenido relacionado como sonido, imagen y demás contenido multimedia. Muchas de ellas registran la actividad del usuario mediante una cuenta, de modo que el aprendizaje es continuo y se queda guardado para un posterior uso.

Nos basaremos en el método flashcard para nuestra aplicación, usando sonido, imagen y palabras en inglés y castellano con registro de actividad del usuario.

LiveMocha: http://livemocha.com/?lang=es

WordSteps: http://wordsteps.com/

Imendi: http://imendi.com/

MemRise: http://www.memrise.com/

Gifted Speech: http://www.giftedspeech.com/

MyLingo: http://es.mylingo.org/

LingQ: http://www.lingq.com/es/

BBC Languages: http://www.bbc.co.uk/languages/

Busuu: http://www.busuu.com/es

Yabla: http://www.yabla.com/

Babbel: http://es.babbel.com/

DuoLingo: http://duolingo.com/es

Verbal Education[11]: http://www.verbaleducation.com/index.aspx

LanguageCourse: http://www.languagecourse.net/

English Vocabulary Flashcards: http://www.saberingles.com.ar/flashcards/

WordReference Java API:
http://www.filebuzz.com/fileinfo/211366/Word_Reference_Java_API.html

7.2 APLICACIONES DE ESCRITORIO

Es el nicho de negocio más abundante de producto software de pago. La mayoría de las aplicaciones vienen en diversos formatos físicos y también disponen ayuda web, y poseen gran contenido didáctico almacenado con una curva de aprendizaje muy moderada y de gran duración.

Las aplicaciones gratuitas de código libre encontradas han sido escasas, con una interfaz más ligera, sencilla y de contenido limitado respecto de las versiones de pago. Hemos intentado adaptar su idea de flashcard, simplificando su estructura monolítica para hacer una aplicación con dos vistas, una para modo Profesor para edición y creación de lecciones e inserción en base de datos local, y otra para el alumno con registro de actividad del usuario por nivel de aprendizaje.

Transparent Language: http://www.transparent.com/personal/tlcompleteedition.html

Tell Me More: http://www.tellmemore.com/homeus.aspx#&panel1-2

Rosseta Stone: http://www.rosettastone.es/

Pimsleur Unlimited: https://www.pimsleurunlimited.com/

Rocket Languages: http://www.rocketlanguages.com/

Mango Passport: http://www.mangolanguages.com/tag/mango-passport/

Fluenz: http://www.fluenz.com/

Living Language Platinum: http://www.livinglanguage.com/

JFlash: http://www.filebuzz.com/fileinfo/205157/flashcards__JFlash_.html

Java Vocabulary Learning Tool (jVLT): http://jvlt.sourceforge.net/

Vocatude: http://oriente-voca.eu/

7.3 APLICACIONES MÓVILES

DuoLingo Iphone App: http://duolingo.com/#!/iphone

KNOWJI TOEFL: http://es.download.cnet.com/Knowji-TOEFL-Audio-Visual-Vocabulary-Flashcards-A-learning-memorization-and-pronunciation-system-with-spaced-repetition/3000-20415_4-75867468.html

KNOWJI PSAT: http://udm4.com/iPhone/Knowji_PSAT_Audio_Vi-4134246

Hello-Hello World: https://itunes.apple.com/us/app/hello-hello-social/id463999669?mt=8

Living Language iPad: https://itunes.apple.com/es/app/living-language-spanish-for/id453350265?mt=8

7.4 SINTETIZADORES DE VOZ DE CÓDIGO LIBRE

Para suplir la carencia y la complejidad de encontrar una base de datos grande de palabras en inglés y habilitar la escalabilidad futura al añadir nuevas palabras, hemos investigado los sintetizadores de voz de código libre para leer texto en inglés. Desgraciadamente, las mejores aplicaciones de pago como Loquendo son de pago, y las de código libre en su mayoría son versiones de más de 5 años y de compleja instalación, tanto del sintetizador de voz como de importación de paquetes de voces en distintos idiomas.

Nos hemos decantado por el uso de la aplicación más extendida y con más ayuda en internet, FreeTTS, que a pesar de tener un repertorio de voces limitado, ha sido la más fácil de configurar y más eficiente en funcionamiento.

Java Speech API: http://www.oracle.com/technetwork/java/jsapifaq-135248.html#spec-available

FreeTTS: http://freetts.sourceforge.net/docs/index.php

MBrola Project: http://tcts.fpms.ac.be/synthesis/mbrola.html

Festvox: http://festvox.org/

The Festival Speech Synthesis System: http://www.cstr.ed.ac.uk/projects/festival/

eSpeak: http://espeak.sourceforge.net/

TalkingJava: http://www.cloudgarden.com/JSAPI/index.html

The MARY Text To Speech: http://mary.dfki.de/

Yakitome!: http://www.yakitome.com/tts/text_to_speech

Spoken Text: http://www.spokentext.net/

Praat: http://www.fon.hum.uva.nl/praat/

Flite: http://www.speech.cs.cmu.edu/flite/

Epos: http://epos.ure.cas.cz/

Loquendo: http://www.loquendo.com/es/

8 CAPITULO I: ESPECIFICACIÓN DE REQUISITOS SOFTWARE

8.1 INTRODUCCION

8.1.1 OBJETIVO

8.1.1.1 PROPÓSITO

Con la especificación de requisitos software (SRS) lo que se pretende es realizar una descripción detallada y precisa de los servicios y restricciones que ofrece nuestro sistema software, así como una descripción de sus funcionalidades.

Para desarrollar la especificación de requisitos software se ha seguido el estándar ISO 830 del IEEE.

8.1.1.2 AUDIENCIA

La audiencia de los SRS hace referencia al cliente con el que se ha establecido el acuerdo para el desarrollo del sistema software. En nuestro caso personal la audiencia coincide con nuestro profesor de la asignatura de Sistemas Informáticos, incluida en la titulación de Ingeniería Informática de la Universidad Complutense de Madrid.

8.1.2 ALCANCE

El proyecto software, con nombre Hello word!, es una aplicación de escritorio implementada en lenguaje Java basada en el Modelo-Vista-Controlador[17].

El objetivo de esta aplicación es enseñar el idioma inglés al usuario de manera fácil y sencilla. Se aplicará el método de aprendizaje mastery learning sobre grados de vocabulario, subdivididos en unidades de bloques de gramática y con refuerzo sobre el histórico del alumno.

8.1.3 APRECIACIÓN GLOBAL

Seguiremos especificando los SRS con el siguiente formato:

- Descripción general: en este punto se describen los requisitos y factores generales que afectan al producto.
  - Perspectiva del producto:
  - Funciones del producto: descripción de las funciones principales del software.
  - Características de usuario: descripción del tipo usuario, incluyendo nivel educacional, experiencia y experiencia técnica.
  - Restricciones: limitaciones del diseño y desarrollo del sistema.
  - Supuestos y dependencias: descripción de factores que afectan a los requisitos especificados en el SRS.
  - Requisitos futuros: descripción de posibles requisitos futuros para el sistema.
- Requisitos específicos: especificación detallada de todos los requisitos que debe cumplir nuestro sistema de desarrollo.
  - Interfaces: descripción detalla de las entradas y salidas del sistema software.
  - Funciones: acciones fundamentales que debe realizar el software al recibir información, procesarla y producir resultados.
  - Requisitos de rendimiento: requisitos de rendimiento del sistema.
  - Requisitos de la base de datos lógica: requisitos de la información almacenada en la base de datos.
  - Restricciones de desarrollo: requisitos a cumplir en el ámbito de desarrollo.

- Atributos del sistema software: atributos que tiene que cumplir el sistema.

8.2 DESCRIPCIÓN GENERAL

8.2.1 PERSPECTIVA DEL PRODUCTO

Nuestro sistema software ha sido desarrollado para la asignatura de Sistemas Informáticos, impartida en la titulación de Ingeniería Informática de la Universidad Complutense de Madrid.

El sistema deberá ser capaz de enseñar el idioma inglés a un usuario de manera fácil, sencilla e intuitiva. Para ello, el sistema guiará al usuario por una serie de niveles y lecciones, donde el usuario puede consultar el contenido de gramática y vocabulario del idioma inglés, y realizar una serie de ejercicios para comprobar el nivel de su aprendizaje. Las lecciones serán explicadas de manera asociativa con imágenes y dispondrá de archivos de sonido para aprender la correcta pronunciación.

El sistema dispondrá de una base de datos por defecto, pero existirá un perfil de profesor, donde éste puede ajustar el contenido de los niveles y lecciones, editándolo o añadiendo lo necesitado para los alumnos.

El sistema será desarrollado en lenguaje Java, debido a los beneficios que dispone su máquina virtual para ejecutarse en distintas plataformas independientemente de su sistema operativo.

La base de datos estará implementada en MySQL, lenguaje sencillo que cubre con las necesidades.

La aplicación será monousuario, por lo que un único usuario puede acceder a la aplicación sin necesidad de credenciales. Aunque, como ya hemos comentado tendrá dos perfiles, el perfil alumno y el perfil profesor con distintos contenidos.

- Interfaces del sistema: El sistema no formará parte de un sistema más grande, será un sistema único.

- Interfaces de usuario: El sistema dispondrá de una interfaz gráfica sencilla y fácil de usar para el usuario.

- Interfaces hardware: La comunicación entre el software y el hardware no necesitará ninguna especificación.

- Interfaces software: La aplicación interaccionará con la base de datos en MySQL, de manera invisible al usuario, para consultar y guardar todos los datos necesarios.

  Nombre: MySQL Community Server

  Versión: 5.6.11

  Enlace de descarga: http://dev.mysql.com/downloads/mysql/

- Interfaces de comunicación: No se requerirá ninguna interfaz de comunicación.

- Memoria: La aplicación requerirá la memoria necesaria para poder almacenar los datos que necesite el programa y poder realizar las operaciones y consultas con rapidez.

- Operaciones: existirán dos perfiles distintos, ambos monousuario, pero con distintas operaciones disponibles dentro de cada perfil. El perfil alumno accederá al contenido y hará uso de él, mientras que el perfil profesor dispondrá de los permisos y opciones necesarias para cambiar el contenido y editarlo para así poder personalizarlo para el alumno.

- Requisitos de adaptación: para que el sistema funcione correctamente, el usuario deberá tener instalada en su máquina y configurados correctamente para el funcionamiento deseado el entorno de ejecución de Java y el gestor de bases de datos, en nuestro caso, MySQL.

8.2.2 FUNCIONES DEL PRODUCTO

El sistema se divide en dos vistas con distintas funcionalidades cada una, la vista del alumno y la vista del profesor.

- La vista del alumno

La vista del alumno va asociada al modo aprendizaje. El alumno tendrá a su disposición en la aplicación una serie de niveles y varias lecciones por nivel.

En cada lección podrá acceder a su contenido, el cual podrá ser más orientado a la gramática del lenguaje y una serie de frases de ejemplo a lo explicado en la sección teórica de gramática, o más orientado a vocabulario, donde podrá ver un listado de palabras con sus correspondientes imágenes.

Además de poder consultar la sección teórica, podrá también acceder a los ejercicios propuestos en cada lección. Según vaya el alumno superando los ejercicios de una lección podrá acceder a los correspondientes ejercicios de la siguiente lección, mientras, permanecerán bloqueados. De esta manera se le obliga al usuario a seguir la metodología correcta.

A su vez el alumno, podrá consultar en la vista perfil, las puntuaciones por niveles y lecciones.

- La vista del profesor

La vista del profesor está pensada para poder editar el contenido de la base de datos. Aunque el sistema ya cuenta con una base de datos, bastante completa, por defecto, se le da la opción a los profesores a editarla a su manera o incluso crear una base de datos nueva para poder explicar el contenido deseado. Por lo que con una fácil interfaz, el profesor puede crear una base de datos desde cero, o desde un fichero, o simplemente editar o añadir más contenido a la base de datos por defecto. En cuanto modifique y guarde estos cambios estarán inmediatamente disponibles para el alumno, el cual no tiene acceso a cambiar nada.

8.2.3 CARACTERÍSTICAS DE USUARIO

El sistema estará enfocado en usuarios con ganas y necesidad de aprender el idioma inglés. Para ello, el usuario debe estar familiarizado con las aplicaciones de escritorio y su debida ejecución. Debido a que el sistema incluirá instrucciones de cómo realizar la instalación necesaria del gestor de base de datos, el usuario no tiene por qué tener conocimiento de éste mismo, pero si estar familiarizado con su sistema operativo para poder realizar la instalación correctamente.

8.2.4 RESTRICCIONES

Las restricciones impuestas para el desarrollo de este proyecto es la normativa impuesta en la asignatura Sistemas Informáticos, impartida en la titulación de Ingeniería Informática de la Universidad Complutense de Madrid.

Dicha normativa puede consultarse en este enlace http://www.fdi.ucm.es/SI/2012-13/P4_normativa_SIaprobada.pdf

Además de la normativa, dispondremos de un conjunto de fechas límite a cumplir:

- Entrega del borrador del proyecto al profesor director: Viernes 7 de junio de 2013
- Entrega del acta de apto/no-apto del profesor al Vicedecano: Lunes 17 de junio de 2013
- Publicación del listado de aptos/no-aptos por parte del Vicedecano: Martes 18 de junio de 2013
- Entrega de la memoria final en secretaría de alumnos: Viernes 21 de junio de 2013
- Exposiciones públicas de todos los alumnos considerados aptos: Del 1 al 3 de julio de 2013
- Publicación de la lista de exposiciones públicas y del tribunal de matrícula de honor: 5 de julio de 2013

- Exposiciones públicas al tribunal de MH: 8 y 9 de julio de 2013

8.2.5 SUPUESTOS Y DEPENDENCIAS

A la hora de definir los requisitos de un proyecto, estos pueden cambiar por parte del cliente y afectar a otros requisitos y suposiciones anteriores. En nuestro caso podemos decir que los requisitos no han sufrido ningún cambio y han sido los definidos desde el principio.

8.2.6 REQUISITOS FUTUROS

En este proyecto como requisitos futuros podemos detallar la ampliación de Idiomas. Ahora mismo la aplicación es específica para el aprendizaje del idioma Inglés, pero con cambios en la base de datos y algún pequeño cambio en código podría ampliarse a más idiomas.

Otro requisito futuro bastante práctico sería la posibilidad de definir la aplicación como multiusuario en vez de como monousuario, de manera que el módulo del alumno funcione con credenciales que identifiquen a cada usuario. Para ello habría que añadir las tablas de alumnos en la base de datos y controlar el acceso, pero de esta manera, muchos usuarios podrían usar una misma aplicación.

8.3 REQUISITOS ESPECÍFICOS

En este apartado se detallarán todos y cada uno de los requisitos que componen y definen nuestro sistema.

8.3.1 INTERFACES

8.3.1.1 INTERFAZ EXTERNA

Definimos como entrada del sistema los datos introducidos desde la base de datos que serán necesarios para su funcionamiento correcto. Estos datos serán introducidos

manualmente por teclado y ratón, o bien desde ficheros de texto. Los tipos de datos definidos en la base de datos serán INTEGER, VARCHAR y BLOB.

8.3.1.2 INTERFAZ USUARIO

La interfaz del usuario tiene que ser clara e intuitiva. Destacará la simplicidad de opciones y posibilidades disponibles, lo importante es que esté todo el proceso de aprendizaje del idioma guiado. El estilo de la interfaz se basa en botones y menús desplegables. No abundarán los textos y los colores serán muy simples y no llamativos.

8.3.2 FUNCIONES

- Módulo alumno

RF001- El usuario alumno solo podrá acceder al contenido y no modificarlo.

RF002- El usuario alumno no podrá crear nuevo contenido en la base de datos.

RF003- El usuario alumno podrá acceder a la vista Aprendizaje y a la vista Usuario.

RF004- En la vista Aprendizaje el usuario seleccionará el nivel y la lección a la que desea acceder.

RF005- En la vista Usuario el usuario puede seleccionar la lección de la que desea saber la puntuación acumulada.

RF006- Una vez seleccionado el nivel y la lección el usuario accederá a una introducción teórica de la lección y un menú: "Ver palabras" y "Ejercicios".

RF007- El botón "Ver palabras" dará acceso a la navegación del contenido de la lección.

RF008- Las lecciones serán de dos tipos: de contenido gramatical y de contenido de vocabulario.

RF009- Si la lección es de contenido gramatical el usuario podrá navegar entre distintas frases en castellano y en inglés.

RF010- Las frases tendrán asociada la opción de escuchar el sonido de su correcta pronunciación.

RF011- Si la lección es de contenido de vocabulario el usuario podrá navegar entre distintas palabras en castellano y en inglés.

RF012- Las palabras tendrán asociadas la opción de escuchar el sonido de su correcta pronunciación.

RF013- Las palabras tendrán asociadas una imagen descriptiva para facilitar su memorización.

RF014- El usuario dispondrá de botones de navegación para ir navegando entre las palabras y frases de una en una hacia delante y hacia atrás.

RF015- El usuario dispondrá de botones de navegación para ir al principio y fin de la navegación de palabras y frases.

RF016- Si el usuario se encuentra consultando la primera frase o palabra, los botones de navegación “inicio” y “anterior” no estarán disponibles.

RF017- Si el usuario se encuentra consultando la última frase o palabra los botones de navegación “fin” y “anterior” no estarán disponibles.

RF018- Solo la palabra o frase en inglés aparecerá en la sección de ejercicios.

RF019- La opción de sonido para escuchar la correcta pronunciación aparecerá junto con la palabra o frase.

RF020- El usuario podrá introducir la correspondiente frase o palabra en español que corresponde con la solución.

RF021- En el momento que el usuario termina de escribir su respuesta aparecerá la respuesta correcta y los botones para que el usuario realice la autocorrección.

RF022- Existirá un botón “Terminar” para guardar y cerrar la pantalla del ejercicio.

RF023- Por cada respuesta acertada se incrementará un contador asociado a la palabra o frase.

RF024- Cuando una palabra o frase se haya acertado tres veces no aparecerá más, se dará por completada.

RF025- Cuando se da una palabra por completada se incrementará la barra de progreso y la puntuación de la lección.

RF026- Si el usuario accede a la pestaña “Usuario” podrá consultar las puntuaciones de cada lección.

- Módulo profesor

RF027- El usuario profesor podrá crear una base de datos nueva en el sistema.

RF028- El usuario profesor podrá elegir entre abrir una base de datos o cargar una base de datos desde un archivo.

RF029- El usuario profesor podrá elegir entre guardar los cambios en base de datos o guardarlos en un archivo.

RF030- El usuario profesor podrá importar una base de datos.

RF031- El usuario profesor podrá exportar los datos de la base de datos.

RF032- El usuario profesor dispondrá de una opción para resetear la base de datos y cargar la base de datos por defecto del sistema.

RF033- En el menú de opciones de edición de la base de datos aparecerán opciones para copiar, pegar, cortar, eliminar, deshacer y seleccionar datos.

RF034- Cuando tenga cargada una base de datos podrá editar manualmente los datos.

RF035- Cuando tenga cargada una base de datos podrá reproducir los sonidos de la correcta pronunciación de palabras y frases del sistema.

RF036- Dispondrá de unas opciones para editar un nivel o una lección una vez cargada una base de datos.

RF037- Dispondrá de unas opciones para poder insertar un nivel, una lección o una palabra a la base de datos cargada.

RF038- Dispondrá de unas opciones para poder borrar un nivel o una lección.

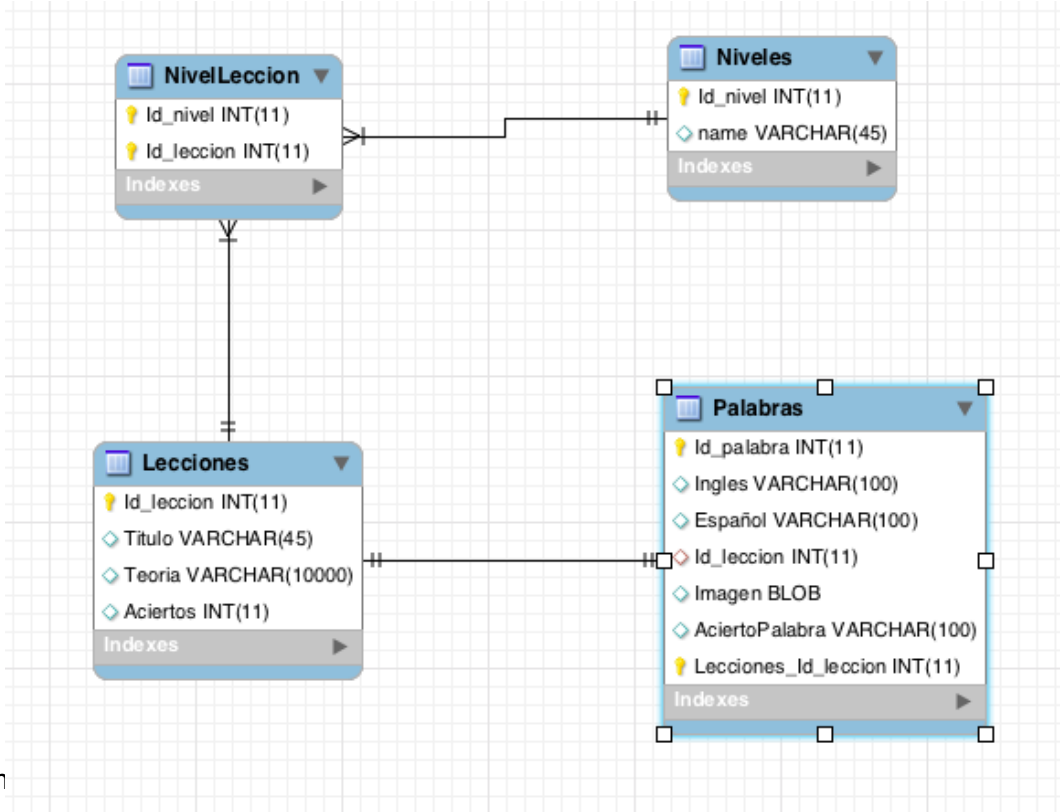
RF039- Dispondrá de una opción para poder salir del sistema.

8.3.3 Requisitos de rendimiento

El sistema deberá de funcionar en los distintos sistemas operativos (Windows, Mac OSX y Linux). Además deberá de tener un tiempo de consultas y modificaciones con la base de datos bastante bajo, inferior a diez segundos. Al no tener gran carga de datos, la aplicación debe funcionar correctamente y no quedarse bloqueada en ningún momento.

8.3.4 Requisitos de base de datos lógica

La base de datos ha sido creada en MySQL. El diseño de la base de datos se muestra en la siguiente imagen.



Com

donde guardaremos toda la información de cada nivel, como el nombre del nivel, el

identificador. Por otro lado tenemos la tabla 'Lecciones' donde guardamos la información correspondiente a cada lección, como su nombre, su identificador, la teoría correspondiente a ésa lección y la puntuación.

La tabla 'NivelLeccion' se encarga de asociar las dos tablas descritas anteriormente.

Por último tenemos la tabla 'Palabras', donde guardaremos la información de cada palabra y frase del sistema. Guardaremos datos como su identificador, la palabra en castellano y la palabra en inglés, el identificador de la lección a la que pertenecen, la imagen correspondiente a cada palabra y la puntuación acumulada por palabra.

8.3.5 RESTRICCIONES DE DESARROLLO

El ciclo de vida elegido para desarrollar el producto será el de prototipo evolutivo orientado a objetos, de manera que se puedan incorporar fácilmente cambios y nuevas funciones, así como aprovechar las ventajas de usabilidad proporcionada por el paradigma de orientación a objetos. El lenguaje para la base de datos será MySQL.

8.3.6 ATRIBUTOS DEL SISTEMA SOFTWARE

Los atributos más destacables del sistema son la velocidad y la estabilidad de las funciones que lo forman. La velocidad, como hemos hablado en otros apartados, es destacable en nuestro sistema debido a su rápida comunicación con la base de datos. Vamos a describir los siguientes atributos del sistema:

- Mantenibilidad: El sistema software deberá ser mantenible y evolucionable. De la manera en la que se ha decidido implementar se podrá añadir más funcionalidades sin problema, por lo que el sistema podrá pasar a formar parte de un sistema mayor en un futuro fácilmente.

- Portabilidad: El sistema software deberá poder ejecutarse en cualquier sistema operativo. Debido al lenguaje en el que se ha decidido implementar la aplicación, Java, se aprovechará el beneficio de su máquina virtual para poder ejecutar la aplicación sin problema en distintos sistemas operativos.

- Estabilidad: El sistema debe ser estable y no producir errores. Para poder conseguir un número mayor de usuarios, el sistema debe ser eficiente y funcionar con agilidad. Un sistema en el que un usuario alumno está realizando un ejercicio y falla implicaría que el usuario alumno tiene que volver a repetir el ejercicio. Por lo que la estabilidad del sistema será de los factores descritos más importantes a cumplir.

- Fiabilidad: El sistema debe funcionar correctamente. El mal funcionamiento del sistema implicaría no aprobar la asignatura de Sistemas Informáticos.

9 CAPITULO II: ARQUITECTURA DE LA APLICACIÓN

9.1 DISEÑO

Con la información presentada en las secciones anteriores de este mismo capítulo se puede proceder al diseño de la aplicación que constituirá el punto de interacción con los usuarios del sistema. A lo largo de la presente sección expondremos los aspectos más relevantes del diseño de la aplicación, que definen gran parte de la implementación que se presenta en el siguiente capítulo.

La estructura de este apartado es la siguiente:

- Modelo Vista Controlador

En este apartado presentamos el patrón de arquitectura del software Modelo-Vista-Controlador, sus principales ventajas y la motivación que nos ha llevado a escogerlo como el patrón que guíe la arquitectura de la aplicación.

- Diseño de la base de datos

En este apartado exponemos el diseño de la base de datos que constituirá el modelo de datos de nuestra aplicación utilizando diagramas de Entidad-Relación.

- Diseño de la interfaz gráfica de usuario

Finalizamos esta sección con unos bocetos del diseño de la interfaz gráfica de usuario que servirán como orientación para su implementación.

9.1.1 MODELO VISTA CONTROLADOR

A la hora de diseñar e implementar una aplicación es altamente recomendable por no decir indispensable aplicar patrones, aunque en un principio la aplicación se prevea sencilla esto nos evitara futuros problemas y nos ayudara en diversos aspectos.

El objetivo de su utilización se puede resumir en la cita de Christopher Alexander:

"Cada patrón describe un problema que ocurre una y otra vez en nuestro entorno, para describir después el núcleo de la solución a ese problema, de tal manera que esa solución pueda ser usada más de un millón de veces, sin hacerlo ni siquiera dos veces de la misma forma"

Christopher Alexander - The Timeless Way of Building

En nuestra aplicación aplicaremos el patrón de arquitectura de aplicación Modelo-Vista-Controlador (MVC). El primer motivo de esta decisión es que todos los miembros del equipo estamos familiarizados con él, debido a asignaturas durante la carrera o a diferentes experiencias laborales. Además, es uno de los patrones más aplicados en las aplicaciones actuales (tanto aplicaciones web como aplicaciones desktop) y en general, todos los frameworks actuales disponen de herramientas que facilitan su implementación y usabilidad.

A continuación explicaremos en qué consiste el Modelo Vista Controlador:

MVC fue introducido por Trygve Reenskaug durante su visita a Xerox Parc en los años 70 y, seguidamente, en los años 80, Jim Althoff y otros implementaron una versión de MVC para la biblioteca de clases de Smalltalk-80. Sólo más tarde, en 1988, MVC se expresó como un concepto general. Este concepto consiste resumidamente en dividir la aplicación en tres componentes principales:

- Modelo: Es la representación específica de la información con la cual el sistema opera, por lo tanto gestiona todos los accesos a dicha información, tanto consultas como actualizaciones, implementando también los privilegios de acceso que se hayan descrito

en las especificaciones de la aplicación (lógica de negocio). Envía a la 'vista' aquella parte de la información que en cada momento se le solicita para que sea mostrada (típicamente a un usuario). Las peticiones de acceso o manipulación de información llegan al 'modelo' a través del 'controlador'.

En nuestra implementación Tenemos una estructura llamada DataCollectionManager que mantiene toda la información necesaria sobre el curso y el usuario. Este objeto se apoya en datos "permanentes" en la base de base de datos MySQL.

- Vista: Presenta el 'modelo' (información y lógica de negocio) en un formato adecuado para interactuar (usualmente la interfaz de usuario) por tanto requiere de dicho 'modelo' la información que debe representar como salida.

En nuestra aplicación usamos los componentes básicos proporcionados por java swing bajo la funcionalidad de cambio de aspecto Look and Feel añadida en java 6 v10. En la sección 9.1.3 explicaremos más detalladamente el diseño e implementación de nuestra interfaz gráfica.

- Controlador: Responde a eventos (usualmente acciones del usuario) e invoca peticiones al 'modelo' cuando se hace alguna solicitud sobre la información (por ejemplo, editar un documento o un registro en una base de datos). También puede enviar comandos a su 'vista' asociada si se solicita un cambio en la forma en que se presenta de 'modelo' (por ejemplo, desplazamiento o scroll por un documento o por los diferentes registros de una base de datos), por tanto se podría decir que el 'controlador' hace de intermediario entre la 'vista' y el 'modelo' (véase Middleware).

En nuestro caso tenemos unas clases con métodos que son llamados desde la vista para operar sobre los datos del DataCollectionManager, así como de devolver a la vista los datos del DataCollectionManager para rellenar los componentes gráficos.

En la figura 1 se describe el flujo de trabajo de nuestra aplicación basado en el patrón Modelo Vista Controlador, y obteniendo los datos desde una fuente externa

(concretamente base de datos, para la parte aprendizaje, y base de datos o fichero para la parte de enseñanza).

En él se aprecia cómo el punto de interacción entre el explorador web del cliente y la aplicación en el servidor es el controlador. También se describe cómo el controlador realiza peticiones parametrizadas al modelo, el cual responde con datos que son enviados, también por el controlador, a la vista. Por último, la vista genera el contenido de la GUI es producida por las clases graficas de java.

Figura 1: Esquema de flujo de trabajo del patrón MVC de nuestra aplicación

9.1.2 DISEÑO DE LA BASE DE DATOS

Debido a la necesidad de acceso distribuido a los datos, y a una posible futura adopción de la arquitectura cliente-servidor en 3 capas, decidimos usar una base de datos para almacenar todos los datos necesarios tanto de cursos como de usuario.

Para ello hemos hecho uso de un Sistema Gestor de Bases de Datos (SGBD). El uso de un SGBD proporciona una serie de ventajas que lo convierten en el sistema más utilizado para cubrir esta necesidad. Por una parte, la tarea de asegurar un buen uso del espacio lógico disponible y de garantizar la integridad de los datos es delegada al SGBD librando al programador de la aplicación de esta responsabilidad. Por otra parte, facilita

el acceso a la información gracias al uso de los denominados lenguajes de consulta de bases de datos y al repertorio de operaciones disponibles.

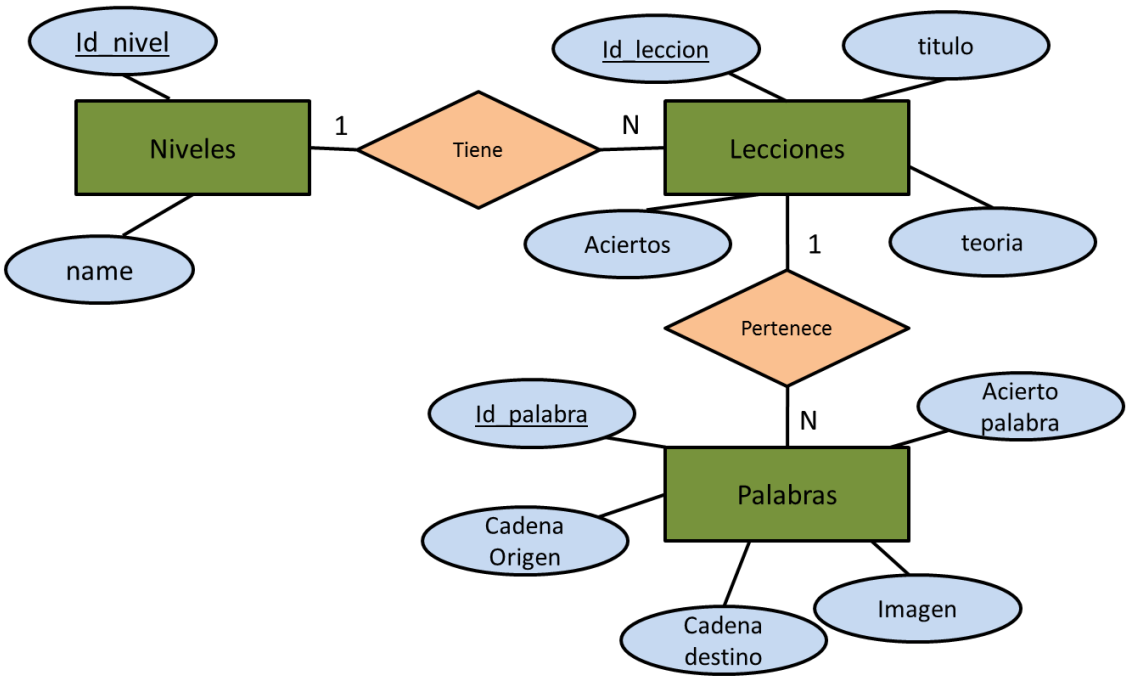


Figura 2: Diagrama Entidad - Relación del modelo de datos de la aplicación

En la figura 2 se representa el diseño del modelo de datos de la aplicación haciendo uso de los diagramas Entidad - Relación. De esta manera podemos definir el modelo de datos sin concretar la tecnología que se va a utilizar para su implementación.

A continuación procedemos a traducir este modelo entidad-relación a modelo de tablas, explicando sus atributos:

- Niveles: Encargada de guardar la información básica de un nivel. Esta sería el identificador único del nivel (id_nivel INT(11)), como clave primaria, y el nombre (name VARCHAR(45)).

- Lecciones: Tabla encargada en guardar la información básica de cada lección. Consta de identificador único de lección (Id_leccion INT(11)) como clave primaria, el título de la lección (Titulo VARCHAR(45)), un campo para la teoría de la lección (Teoría VARCHAR(45)), y un entero para guardar el número de aciertos de dicha lección (Acierto INT(11)), y con este poder gestionar el funcionamiento del curso.

➢ NivelLecciones: Es la relación entre un nivel y todas las lecciones dependientes de este. Para ello usamos una doble clave primaria, donde relacionamos identificador de nivel (Id_nivel INT(11)), con identificador de lección (Id_leccion INT(11)).

➢ Palabras: Contiene las tuplas de palabras o de frases en los idiomas origen-destino. Primero, un identificador único para la tupla, que actúa como clave primaria (Id_palabra INT(11)), un campo para la cadena del idioma origen (Ingles VARCHAR INT(11)), otro para la cadena del idioma destino (Español VARCHAR(11)), un identificador de la lección a la cual pertenece la tupla (Id_leccion INT(11)), una campo para guardar una imagen representativa (Imagen BLOB) y por ultimo otro para almacenar el las veces que se ha acertado esa tupla (AciertoPalabra VARCHAR(100)).

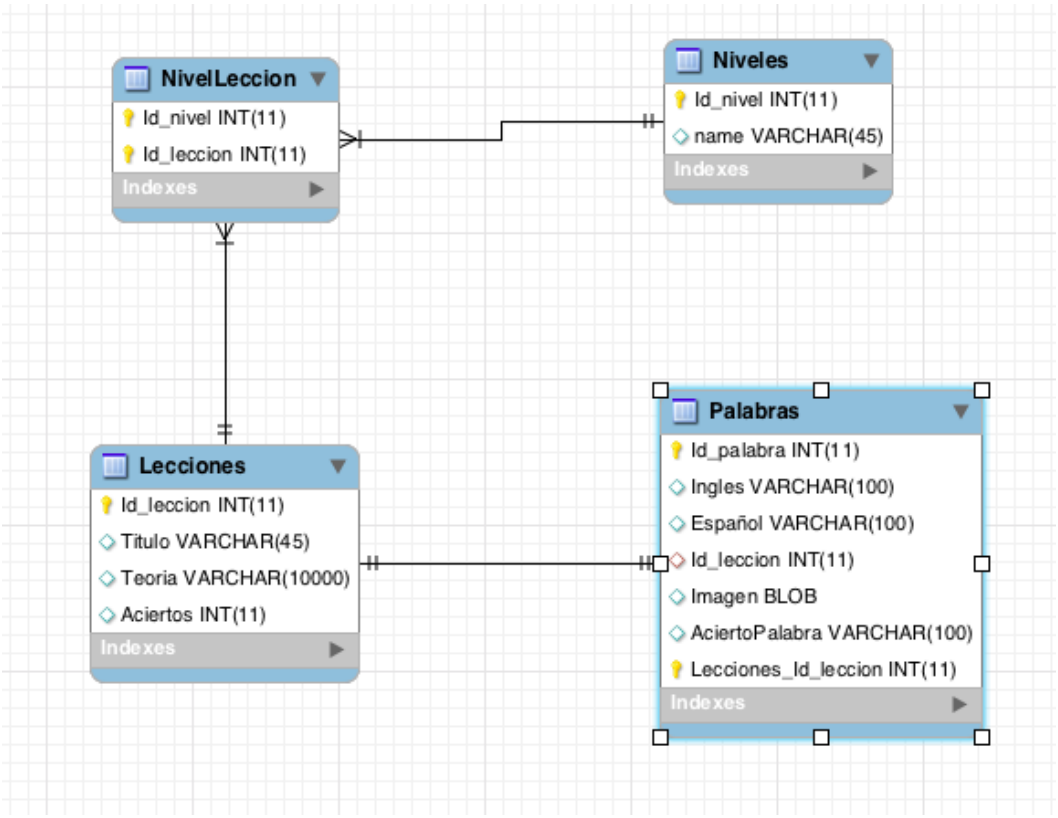


Figura 3: EEE (Enhanced entity-relationship) del modelo de datos de la aplicación

9.1.3 DISEÑO DE LA INTERFAZ GRÁFICA DE USUARIO

Se denomina Interfaz Gráfica de Usuario (GUI) al punto de interacción de un usuario con un sistema informático mediante una presentación gráfica de la información y de los controles. Uno de los criterios de calidad de software que ha cobrado mayor importancia en los últimos años es la usabilidad, concepto definido por la ISO 4 de la siguiente manera:

"La usabilidad se refiere a la capacidad de un software de ser comprendido, aprendido, usado y ser atractivo para el usuario, en condiciones específicas de uso ISO/IEC 9126"

Dado que uno de nuestros objetivos es facilitar el aprendizaje de vocabulario y estructuras gramaticales de idiomas por parte del alumno y la creación y modificación de cursos por parte del profesor, una interfaz usable y accesible es un factor determinante para el logro de nuestros objetivos.

A continuación describiremos el diseño general de la GUI de nuestro sistema haciendo uso de bocetos, prescindiendo de los elementos estéticos.

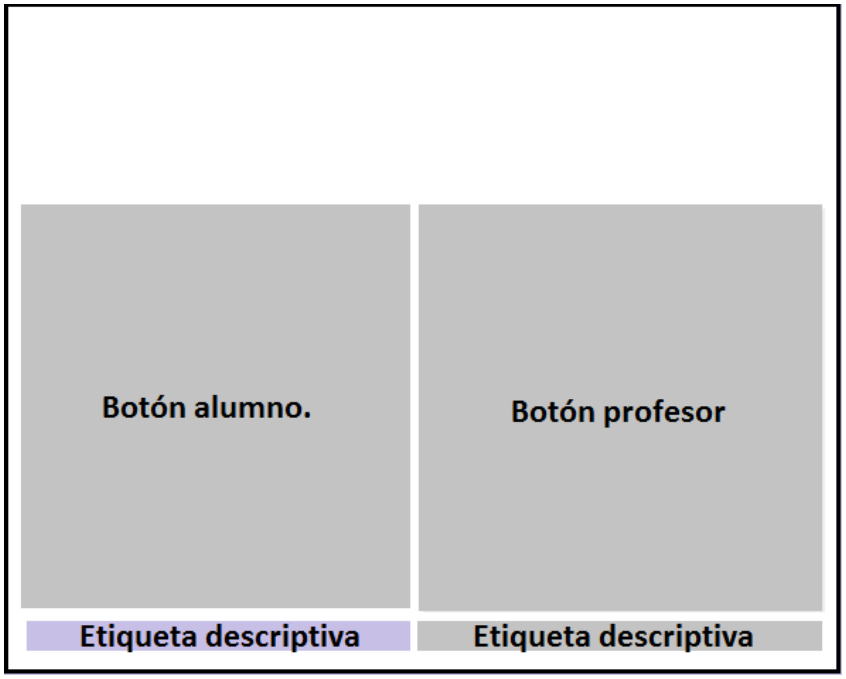


Figura 4: Diseño de la vista principal

- Vista principal:

La ventana principal, nos permite elegir entrar en la vista del alumno o en la vista del profesor. En lo funcional, simplemente tiene dos grandes botones para hacer esa elección.

  - Botón alumno: Carga la ventana de la vista del alumno.
  - Botón profesor: Carga la ventana de la vista del profesor.

- Vista alumno:

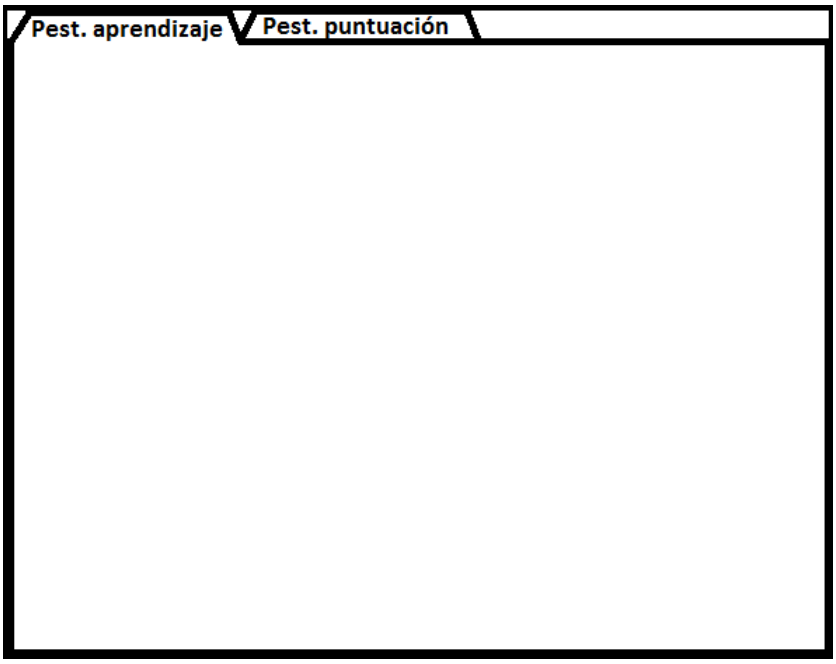


Figura 5: Diseño de la vista alumno

En la vista de alumno nos encontramos con una división entre la vista para el aprendizaje y la vista para la información sobre el usuario alumno, donde se muestra las puntuaciones de las lecciones. Para esta división usaremos un sistema de pestañas.

- Pestaña aprendizaje:

Figura 6: Diseño de la vista aprendizaje.

La pestaña de aprendizaje, nos muestra un panel donde poder elegir mediante menús desplegables, el nivel y la lección que queremos trabajar, y un botón para validar la selección. En la parte inferior se sitúa un panel donde interactuaremos con todas las funciones de la lección como estudiar la teoría, aprender palabras o estructuras gramaticales, o hacer ejercicios.

- Panel principal (panel de lección - Teoría)

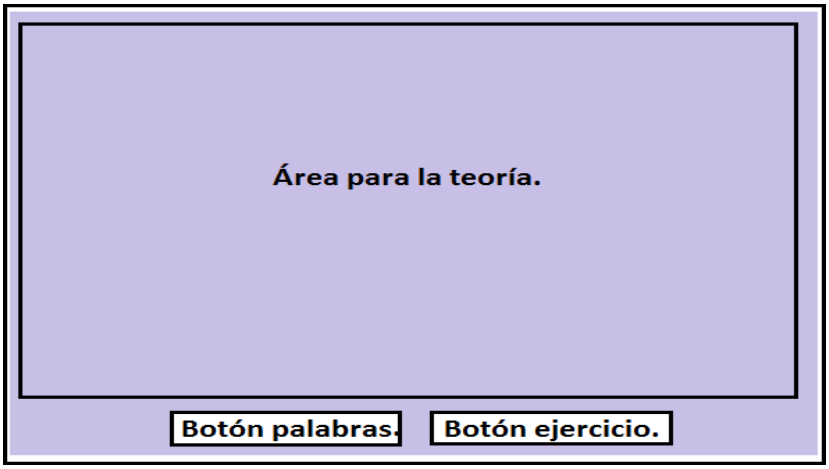


Figura 7: Diseño de la vista del panel de lección.

Desde esta vista de teoría, podremos visualizar la teoría de la lección trabajada, y navegar hasta la vista de palabras (aprendizaje palabra a palabra), o a la vista de ejercicio.

- Vista palabras (panel de lección – Palabras):

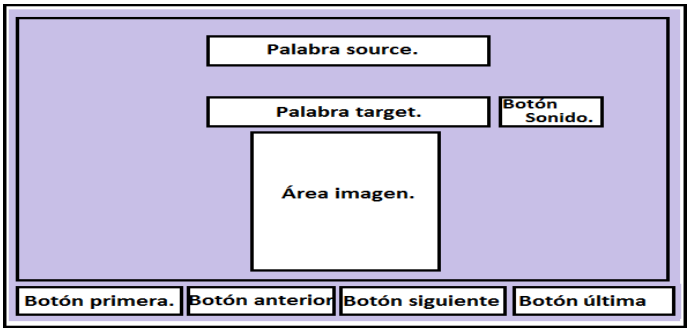


Figura 8: Diseño de la vista palabras

Sobre la vista para el estudio de las palabras de la lección, se basa en un panel superior específico por registro de palabra, y un panel inferior con botones común para todas las palabras.

El panel superior muestra la información sobre un registro, la cadena origen, la cadena destino junto con un botón para reproducir la pronunciación, y seguido de una imagen representativa. Sobre el panel inferior para navegar, este nos permite movernos de registro en registro con los botones con significado trivial. El botón primera y ultima, deberán ser visibles o no según se requiera.

- Vista ejercicio (panel de lección – Ejercicio):

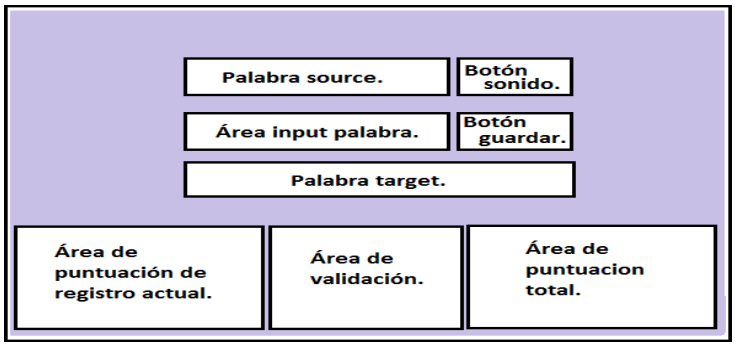


Figura 9: Diseño de la vista ejercicio

El panel superior muestra la información sobre un registro, la cadena origen, la cadena destino junto con un botón para reproducir la pronunciación, y seguido de una imagen representativa. Sobre el panel inferior para navegar, este nos permite movernos de registro en registro con los botones con significado trivial. El botón primera y última, deberán ser visibles o no según se requiera.

- Pestaña puntuación:

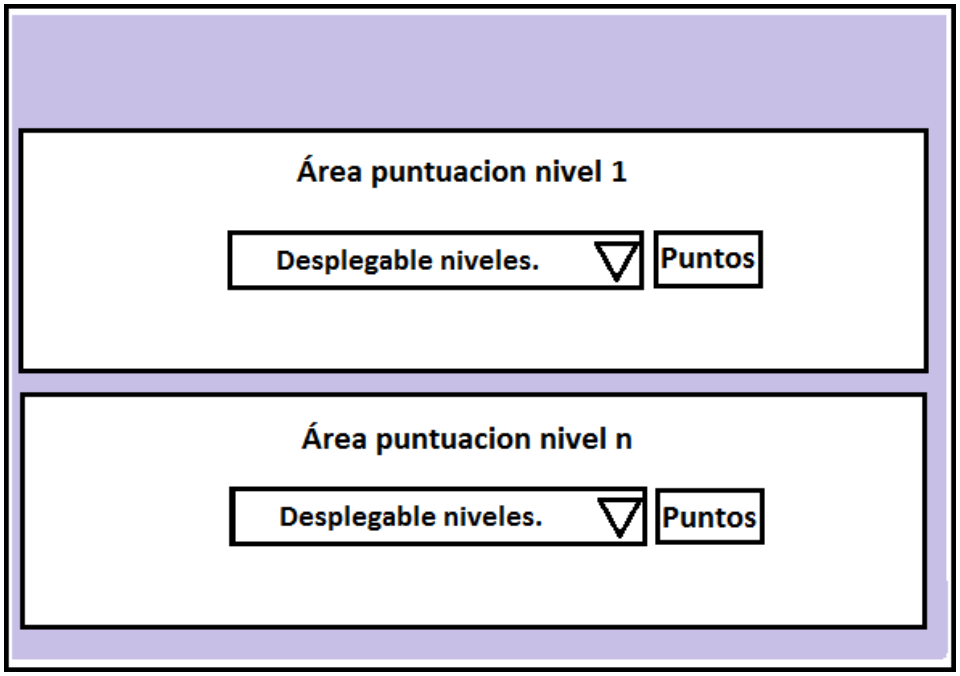


Figura 10: Diseño de la vista puntuación.

En la vista de la pestaña de puntuación, veremos todos los niveles, con un menú desplegable en cada uno para seleccionar lección y así ver la puntuación del usuario en esta.

- Vista profesor:

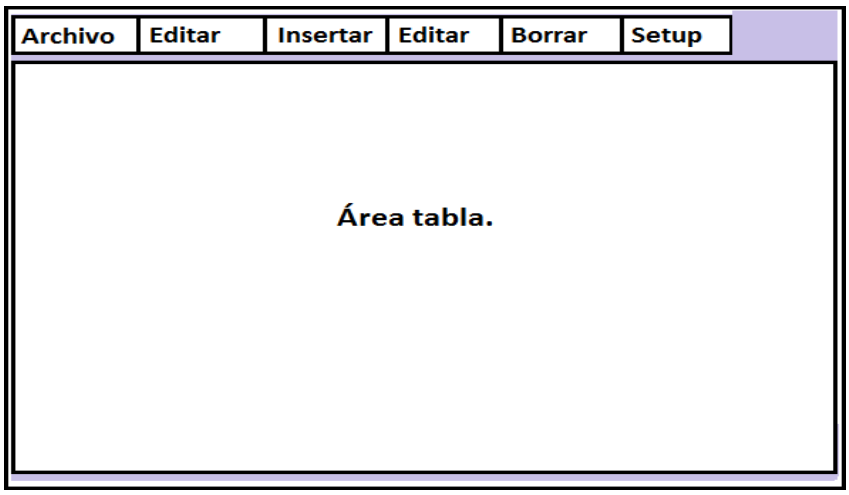


Figura 11: Diseño de la vista profesor.

En la vista de profesor tenemos en la parte superior un menú con múltiples opciones, y en la central una tabla donde se cargaran todas las entradas del curso. Desde insertar, editar o borrar accederemos a un nuevo formulario para desempeñar esas funciones.

Estos formularios todos siguen el siguiente patrón de ventana:

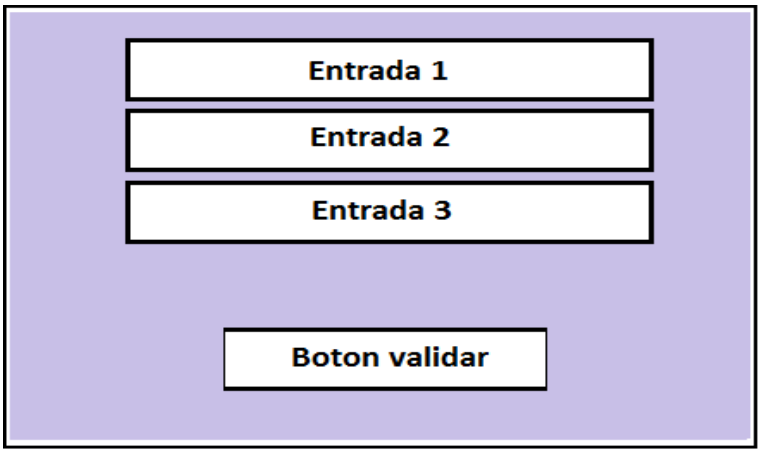


Figura 12: Diseño de la vista formulario.

En la parte superior aparecerán los componentes de entrada a rellenar (o si estamos modificando saldrán rellenos para permitirnos cambiarlos), y en la parte inferior un botón para confirmar la operación.

9.2 IMPLEMENTACIÓN

En este apartado expondremos el diseño y la implementación de la aplicación que hemos elaborado a partir de los requisitos y las especificaciones que se recogen en los apartados anteriores. El objetivo de esta sección es proporcionar al lector una visión clara y general de los aspectos más relevantes del diseño, así como introducir las tecnologías que se utilizarán para su implementación y justificar su elección.

La estructura de este apartado es la siguiente:

- Tecnologías y lenguajes de programación: en este apartado especificaremos las tecnologías que se utilizarán en la implementación de la aplicación, los lenguajes de programación que emplearemos y los motivos que nos han llevado a tomar estas decisiones.
    - Java
    - MySQL
- Modelo de datos: aquí se describe la implementación de la capa de Modelo.
    - Librerías de acceso a datos
    - DataCollectionManager (Gestor del objeto curso)
- Vistas: aquí trataremos la capa de Vista.
- Controladores: en esta sección se describe la capa de Controlador.
    - Otras librerías usadas. Aquí explicaremos más detalladamente el principal funcionamiento de las librerías externas a java que hemos utilizado y las funcionalidades que ellas nos han aportado.
        - Sintetizador de voz: FreeTTS
        - LookAndFeel: Lipstik
    - Patrones de diseño[21]. En este apartado explicaremos que es un patrón de diseño y cuales hemos usado justificando las razones.
        - Singleton [GOF95]
        - Iterator [GOF95]

9.2.1 TECNOLOGÍAS Y LENGUAJES DE PROGRAMACIÓN

En primer lugar comenzaremos definiendo las tecnologías y lenguajes de programación que vamos a utilizar para la implementación del programa Hello Word!, justificando las decisiones tomadas por el equipo.

Las opciones que barajamos para la implementación del programa fueron C++, .net y java. Descartamos C++ porque hacía años que no hacíamos uso profundo con este lenguaje y nos produjo un cierto rechazo generalizado por la incomodidad de algunos de los métodos para este lenguaje comparado con los otros dos y sus framework actualizados a la última constantemente.

Sobre .net los conocimientos que teníamos eran todos externos a la carrera, y no muy amplios así que deberíamos proceder primero a una fase de aprendizaje. Además nuestro deseo de que el software fuese multiplataforma, sin complicaciones en las configuraciones, y a la necesidad del desarrollo tanto desde plataforma Windows como Mac, nos hizo decantarnos por java, que tiene unas herramientas de desarrollo muy pulidas, extendidas y homogéneas para cualquiera de las plataformas. A todo esto sumar nuestro conocimiento suficientemente amplio sobre java aprendido en la carrera y en diversas actividades externas (cursos, trabajo, etc.)

El sistema gestor de base de datos que hemos utilizado es MySQL. Las diferentes opciones que barajamos fueron MySQL, Oracle, Microsoft Access y archivos XML. Los XML los descartamos por ser un método un tanto rudimentario y Microsoft Access por la dificultad que suponía en el desarrollo y uso desde Mac y Linux. En cuanto a decantarnos por MySQL o Oracle fue porque el lenguaje de consulta utilizado por estos sistemas es SQL, lenguaje que conocemos todos los integrantes del grupo y con el que estamos familiarizado, en segundo lugar por ser multiplataforma y tener buen soporte tanto en Windows como en Mac y además con una gran comunidad que los usa, documenta y publica problemas de uso y sus soluciones. Desequilibramos la balanza entre estos dos hacia MySQL por ser más ligero y por estar publicada bajo una licencia de código libre.

9.2.1.1 JAVA

Java es un lenguaje de programación originalmente desarrollado por James Gosling de Sun Microsystems (la cual fue adquirida por la compañía Oracle) y publicado en el 1995 como un componente fundamental de la plataforma Java de Sun Microsystems. El lenguaje deriva mucho de su sintaxis de C y C++, pero tiene menos facilidades de bajo nivel que cualquiera de ellos. Las aplicaciones de Java pueden correr generalmente en cualquier máquina virtual Java (JVM) sin importar la arquitectura de la computadora. Java es un lenguaje de programación de propósito general, concurrente, basado en clases, y orientado a objetos, que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Java es, a partir del 2012, uno de los lenguajes de programación más populares en uso, particularmente para aplicaciones de cliente-servidor de web, con unos 10 millones de usuarios reportados.1 2

La compañía Sun desarrolló la implementación de referencia original para los compiladores de Java, máquinas virtuales, y librerías de clases en 1991 y las publicó por primera vez en el 1995. A partir de mayo del 2007, en cumplimiento con las especificaciones del Proceso de la Comunidad Java, Sun volvió a licenciar la mayoría de sus tecnologías de Java bajo la Licencia Pública General de GNU. Otros también han desarrollado implementaciones alternas a estas tecnologías de Sun, tales como el Compilador de Java de GNU y el GNU Classpath.

El lenguaje Java se creó con cinco objetivos principales:

- Debería usar el paradigma de la programación orientada a objetos.
- Debería permitir la ejecución de un mismo programa en múltiples sistemas operativos.
- Debería incluir por defecto soporte para trabajo en red.
- Debería diseñarse para ejecutar código en sistemas remotos de forma segura.
- Debería ser fácil de usar y tomar lo mejor de otros lenguajes orientados a objetos, como C++.

❖ Orientado a objetos

La primera característica, orientado a objetos ("OO"), se refiere a un método de programación y al diseño del lenguaje. Aunque hay muchas interpretaciones para OO, una primera idea es diseñar el software de forma que los distintos tipos de datos que usen estén unidos a sus operaciones. Así, los datos y el código (funciones o métodos) se combinan en entidades llamadas objetos. Un objeto puede verse como un paquete que contiene el "comportamiento" (el código) y el "estado" (datos). El principio es separar aquello que cambia de las cosas que permanecen inalterables. Frecuentemente, cambiar una estructura de datos implica un cambio en el código que opera sobre los mismos, o viceversa. Esta separación en objetos coherentes e independientes ofrece una base más estable para el diseño de un sistema software. El objetivo es hacer que grandes proyectos sean fáciles de gestionar y manejar, mejorando como consecuencia su calidad y reduciendo el número de proyectos fallidos. Otra de las grandes promesas de la programación orientada a objetos es la creación de entidades más genéricas (objetos) que permitan la reutilización del software entre proyectos, una de las premisas fundamentales de la Ingeniería del Software. Un objeto genérico "cliente", por ejemplo, debería en teoría tener el mismo conjunto de comportamiento en diferentes proyectos, sobre todo cuando estos coinciden en cierta medida, algo que suele suceder en las grandes organizaciones. En este sentido, los objetos podrían verse como piezas reutilizables que pueden emplearse en múltiples proyectos distintos, posibilitando así a la industria del software a construir proyectos de envergadura empleando componentes ya existentes y de comprobada calidad; conduciendo esto finalmente a una reducción drástica del tiempo de desarrollo. Podemos usar como ejemplo de objeto el aluminio. Una vez definidos datos (peso, maleabilidad, etc.), y su "comportamiento" (soldar dos piezas, etc.), el objeto "aluminio" puede ser reutilizado en el campo de la construcción, del automóvil, de la aviación, etc.

La reutilización del software ha experimentado resultados dispares, encontrando dos dificultades principales: el diseño de objetos realmente genéricos es pobremente comprendido, y falta una metodología para la amplia comunicación de oportunidades de reutilización. Algunas comunidades de "código abierto" (open source) quieren ayudar en este problema dando medios a los desarrolladores para diseminar la información sobre el

uso y versatilidad de objetos reutilizables y bibliotecas de objetos.

- Independencia de la plataforma

La segunda característica, la independencia de la plataforma, significa que programas escritos en el lenguaje Java pueden ejecutarse igualmente en cualquier tipo de hardware. Este es el significado de ser capaz de escribir un programa una vez y que pueda ejecutarse en cualquier dispositivo, tal como reza el axioma de Java, ‘’’write once, run anywhere’’’.

Para ello, se compila el código fuente escrito en lenguaje Java, para generar un código conocido como “bytecode” (específicamente Java bytecode)—instrucciones máquina simplificadas específicas de la plataforma Java. Esta pieza está “a medio camino” entre el código fuente y el código máquina que entiende el dispositivo destino. El bytecode es ejecutado entonces en la máquina virtual (JVM), un programa escrito en código nativo de la plataforma destino (que es el que entiende su hardware), que interpreta y ejecuta el código. Además, se suministran bibliotecas adicionales para acceder a las características de cada dispositivo (como los gráficos, ejecución mediante hebras o threads, la interfaz de red) de forma unificada. Se debe tener presente que, aunque hay una etapa explícita de compilación, el bytecode generado es interpretado o convertido a instrucciones máquina del código nativo por el compilador JIT (Just In Time).

Hay implementaciones del compilador de Java que convierten el código fuente directamente en código objeto nativo, como GCJ. Esto elimina la etapa intermedia donde se genera el bytecode, pero la salida de este tipo de compiladores sólo puede ejecutarse en un tipo de arquitectura.

La licencia sobre Java de Sun insiste que todas las implementaciones sean “compatibles”. Esto dio lugar a una disputa legal entre Microsoft y Sun, cuando éste último alegó que la implementación de Microsoft no daba soporte a las interfaces RMI y JNI además de haber añadido características ‘’dependientes’’ de su plataforma. Sun demandó a Microsoft y ganó por daños y perjuicios (unos 20 millones de dólares) así como una orden judicial forzando la acatación de la licencia de Sun. Como respuesta,

Microsoft no ofrece Java con su versión de sistema operativo, y en recientes versiones de Windows, su navegador Internet Explorer no admite la ejecución de applets sin un conector (o plugin) aparte. Sin embargo, Sun y otras fuentes ofrecen versiones gratuitas para distintas versiones de Windows.

Las primeras implementaciones del lenguaje usaban una máquina virtual interpretada para conseguir la portabilidad. Sin embargo, el resultado eran programas que se ejecutaban comparativamente más lentos que aquellos escritos en C o C++. Esto hizo que Java se ganase una reputación de lento en rendimiento. Las implementaciones recientes de la JVM dan lugar a programas que se ejecutan considerablemente más rápido que las versiones antiguas, empleando diversas técnicas, aunque sigue siendo mucho más lento que otros lenguajes.

La primera de estas técnicas es simplemente compilar directamente en código nativo como hacen los compiladores tradicionales, eliminando la etapa del bytecode. Esto da lugar a un gran rendimiento en la ejecución, pero tapa el camino a la portabilidad. Otra técnica, conocida como compilación JIT (Just In Time, o ‘’’compilación al vuelo’’’), convierte el bytecode a código nativo cuando se ejecuta la aplicación. Otras máquinas virtuales más sofisticadas usan una “recompilación dinámica” en la que la VM es capaz de analizar el comportamiento del programa en ejecución y recompila y optimiza las partes críticas. La recompilación dinámica puede lograr mayor grado de optimización que la compilación tradicional (o estática), ya que puede basar su trabajo en el conocimiento que de primera mano tiene sobre el entorno de ejecución y el conjunto de clases cargadas en memoria. La compilación JIT y la recompilación dinámica permiten a los programas Java aprovechar la velocidad de ejecución del código nativo sin por ello perder la ventaja de la portabilidad en ambos.

La portabilidad es técnicamente difícil de lograr, y el éxito de Java en ese campo ha sido dispar. Aunque es de hecho posible escribir programas para la plataforma Java que actúen de forma correcta en múltiples plataformas de distinta arquitectura, el gran número de estas con pequeños errores o inconsistencias llevan a que a veces se parodie el eslogan de Sun, "Write once, run anywhere" como "Write once, debug everywhere" (o “Escríbelo una vez, ejecútalo en cualquier parte” por “Escríbelo una vez, depúralo en todas partes”)

El concepto de independencia de la plataforma de Java cuenta, sin embargo, con un gran éxito en las aplicaciones en el entorno del servidor, como los Servicios Web, los Servlets, los Java Beans, así como en sistemas empotrados basados en OSGi, usando entornos Java empotrados.

- El recolector de basura

En Java el problema fugas de memoria se evita en gran medida gracias a la recolección de basura (o automatic garbage collector). El programador determina cuándo se crean los objetos y el entorno en tiempo de ejecución de Java (Java runtime) es el responsable de gestionar el ciclo de vida de los objetos. El programa, u otros objetos pueden tener localizado un objeto mediante una referencia a éste. Cuando no quedan referencias a un objeto, el recolector de basura de Java borra el objeto, liberando así la memoria que ocupaba previniendo posibles fugas (ejemplo: un objeto creado y únicamente usado dentro de un método sólo tiene entidad dentro de éste; al salir del método el objeto es eliminado). Aun así, es posible que se produzcan fugas de memoria si el código almacena referencias a objetos que ya no son necesarios—es decir, pueden aún ocurrir, pero en un nivel conceptual superior. En definitiva, el recolector de basura de Java permite una fácil creación y eliminación de objetos y mayor seguridad.

9.2.1.2 MYSQL

MySQL es un sistema de gestión de bases de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones.1 MySQL AB —desde enero de 2008 una subsidiaria de Sun Microsystems y ésta a su vez de Oracle Corporation desde abril de 2009— desarrolla MySQL como software libre en un esquema de licenciamiento dual.

Por un lado se ofrece bajo la GNU GPL para cualquier uso compatible con esta licencia, pero para aquellas empresas que quieran incorporarlo en productos privativos deben comprar a la empresa una licencia específica que les permita este uso. Está desarrollado en su mayor parte en ANSI C.

Al contrario de proyectos como Apache, donde el software es desarrollado por una comunidad pública y los derechos de autor del código están en poder del autor individual, MySQL es patrocinado por una empresa privada, que posee el copyright de la mayor parte del código.

Esto es lo que posibilita el esquema de licenciamiento anteriormente mencionado. Además de la venta de licencias privativas, la compañía ofrece soporte y servicios. Para sus operaciones contratan trabajadores alrededor del mundo que colaboran vía Internet. MySQL AB fue fundado por David Axmark, Allan Larsson y Michael Widenius.

❖ Características

Inicialmente, MySQL carecía de elementos considerados esenciales en las bases de datos relacionales, tales como integridad referencial y transacciones. A pesar de ello, atrajo a los desarrolladores de páginas web con contenido dinámico, justamente por su simplicidad. Poco a poco los elementos de los que carecía MySQL están siendo incorporados tanto por desarrollos internos, como por desarrolladores de software libre.

Entre las características disponibles en las últimas versiones se puede destacar:

1. Amplio subconjunto del lenguaje SQL. Algunas extensiones son incluidas igualmente.
2. Disponibilidad en gran cantidad de plataformas y sistemas.
3. Posibilidad de selección de mecanismos de almacenamiento que ofrecen diferente velocidad de operación, soporte físico, capacidad, distribución geográfica, transacciones...
4. Transacciones y claves foráneas.
5. Conectividad segura.
6. Replicación.
7. Búsqueda e indexación de campos de texto.

MySQL es un sistema de administración de bases de datos. Una base de datos es una colección estructurada de tablas que contienen datos. Esta puede ser desde una simple lista de compras a una galería de pinturas o el vasto volumen de información en

una red corporativa. Para agregar, acceder a y procesar datos guardados en un computador, usted necesita un administrador como MySQL Server. Dado que los computadores son muy buenos manejando grandes cantidades de información, los administradores de bases de datos juegan un papel central en computación, como aplicaciones independientes o como parte de otras aplicaciones.

MySQL es un sistema de administración relacional de bases de datos. Una base de datos relacional archiva datos en tablas separadas en vez de colocar todos los datos en un gran archivo. Esto permite velocidad y flexibilidad. Las tablas están conectadas por relaciones definidas que hacen posible combinar datos de diferentes tablas sobre pedido.

MySQL es software de fuente abierta. Fuente abierta significa que es posible para cualquier persona usarlo y modificarlo. Cualquier persona puede bajar el código fuente de MySQL y usarlo sin pagar. Cualquier interesado puede estudiar el código fuente y ajustarlo a sus necesidades. MySQL usa el GPL (GNU General Public License) para definir qué puede hacer y qué no puede hacer con el software en diferentes situaciones. Si usted no se ajusta al GPL o requiere introducir código MySQL en aplicaciones comerciales, usted puede comprar una versión comercial licenciada.

9.2.2 MODELO DE DATOS

El modelo de datos de nuestra aplicación se puede dividir en dos tipos, los encargados de gestionar la memoria persistente, y los encargados de gestionar la memoria volátil.

Por memoria persistente, nos referimos a la capacidad de un dato u objeto para seguir existiendo tras determinadas operaciones, en nuestro caso, a la operación de cerrar la aplicación. Estos datos permanecerán existiendo en el tiempo, permitiéndonos así tener un curso establecido para que el alumno pueda trabajar sobre él, o el profesor hacer modificaciones.

Por memoria volátil, nos referimos información se pierde al interrumpirse una operación, en nuestro caso al cerrar la aplicación. Sería toda la memoria almacenada en

forma de objetos java, en la memoria RAM del pc. Todos estos datos se perderán al cerrar la aplicación, y si queremos que perduren deberemos proporcionar funciones para convertirlos en memoria persistente.

9.2.2.1 MODELO DE DATOS PARA MEMORIA PERSISTENTE

En nuestro caso, nos interesa guardar el curso de forma que este accesible en cualquier momento, ya sea para trabajar sobre el (alumno) o para editarlo (profesor). Para este propósito hemos utilizado la base de datos, donde estos datos permanecerán sin depender del estado de la aplicación java.

Otro caso concreto para el profesor, es la posibilidad de guardar el curso en un fichero, y así poder trasladarlo de modo seguro, enviarlo o simplemente para hacer una copia de seguridad y este poder aplicarlo en cualquier momento futuro, cargándolo en la base de datos, y haciéndolo efectivo sobre el curso que deberá superar el alumno. Para este tipo de datos persistentes, usamos ficheros binarios.

9.2.2.2 MODELO DE DATOS PARA MEMORIA VOLÁTIL

Esta capa del modelo de datos es una intermedia entre la base de datos, y los operadores (métodos java), sobre estos datos. En otros lenguajes o tecnologías, es necesario operar "directamente" sobre los datos de la base de datos, pero en java esto no es una buena práctica (retardos, cuelgues, lógica complicada en base de datos). Lo que hemos realizado ha sido la técnica mapeo objeto-relacional (ORM), hay algunas herramientas muy avanzadas para solventar este problema, como el uso de hibernate[20] (disponible para java y .net), pero en nuestro caso con la necesidad de una base de datos extremadamente sencilla, y un modelo de datos de la aplicación no demasiado complicado, hemos optado por hacer el mapeo objeto-relacional sin el uso de estas herramientas, codificando nosotros los objetos y operaciones.

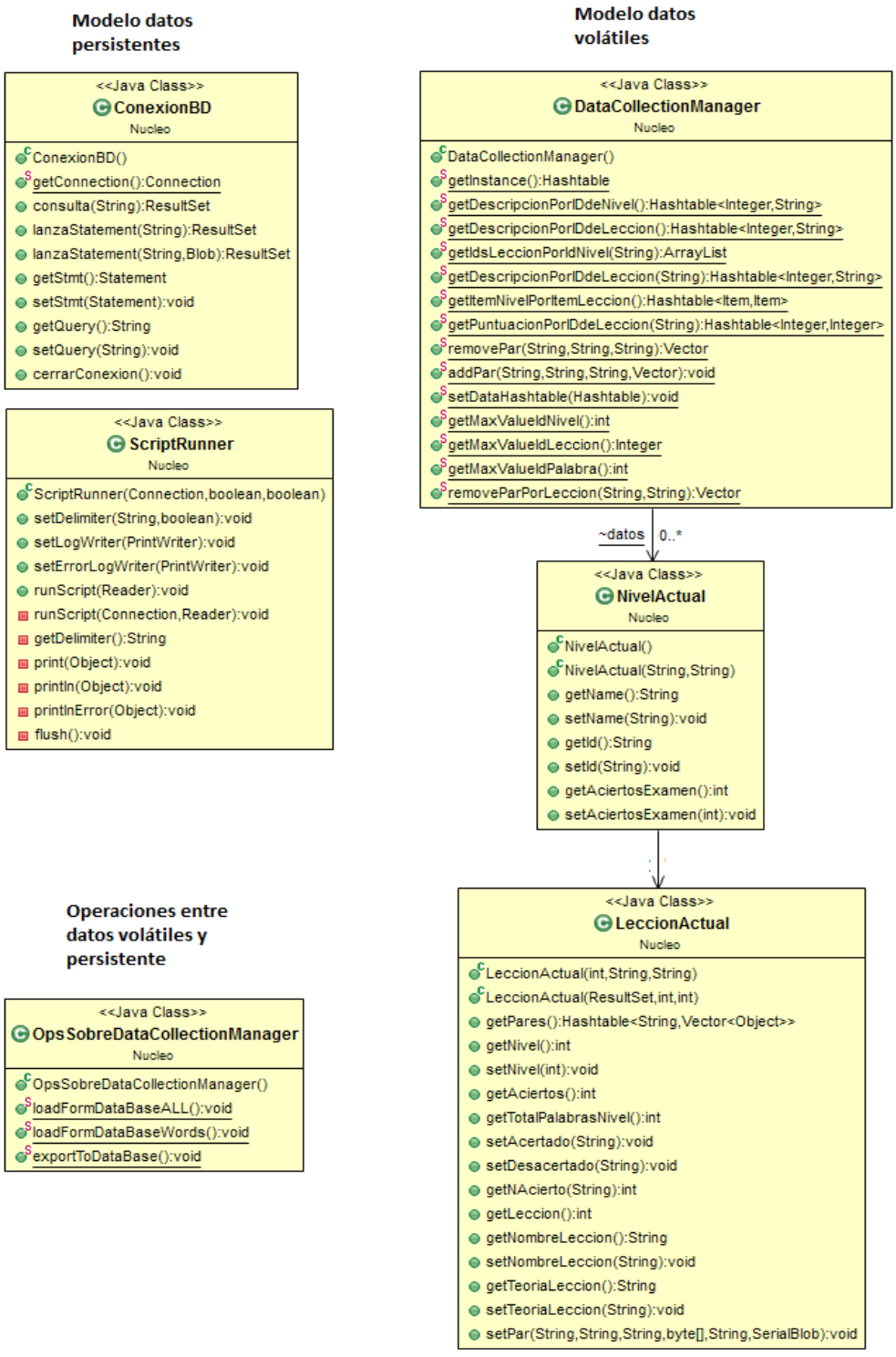


Figura 13: Diagrama UML del modelo

En la Figura 13 se muestra el diagrama UML de la parte del modelo, donde podemos observar unas clases destinadas a gestionar la memoria de datos persistente (lo referente a la base de datos), otras clases destinadas a gestionar la memoria de datos volátil (lo referente con el objeto java que mantiene viva la información) y una clases para operaciones que nos faciliten el intercambio de información entre los dos modelos de memoria.

ConexionDB nos facilita toda la comunicación (apertura de conexión, ejecución de sentencias, cierre de conexión) con sentencias SQL sobre nuestro SGBD de MYSQL. La clase scriptRunner nos permite ejecutar un fichero de texto plano con sentencias SQL sobre la base de datos de forma automática, permitiéndonos así una rápida configuración (creación de tablas y carga de datos) de nuestra base de datos para poder empezar a trabajar desde la vista profesor o incluso desde la vista alumno.

DataCollectionManager es un objeto que representa la entidad de curso. Todos los datos necesarios de un curso están presentes en este objeto que se rige por una serie de Hashtables anidados, terminando en un objeto Vector que contiene todos los atributos de un registro de palabras (cadena A, cadena B, imagen, numero de aciertos, etc.).

En el siguiente diagrama podemos observar la estructura del objeto DataCollectionManager, ordenado por nivel, posteriormente por lección y por ultimo por registro de palabra.

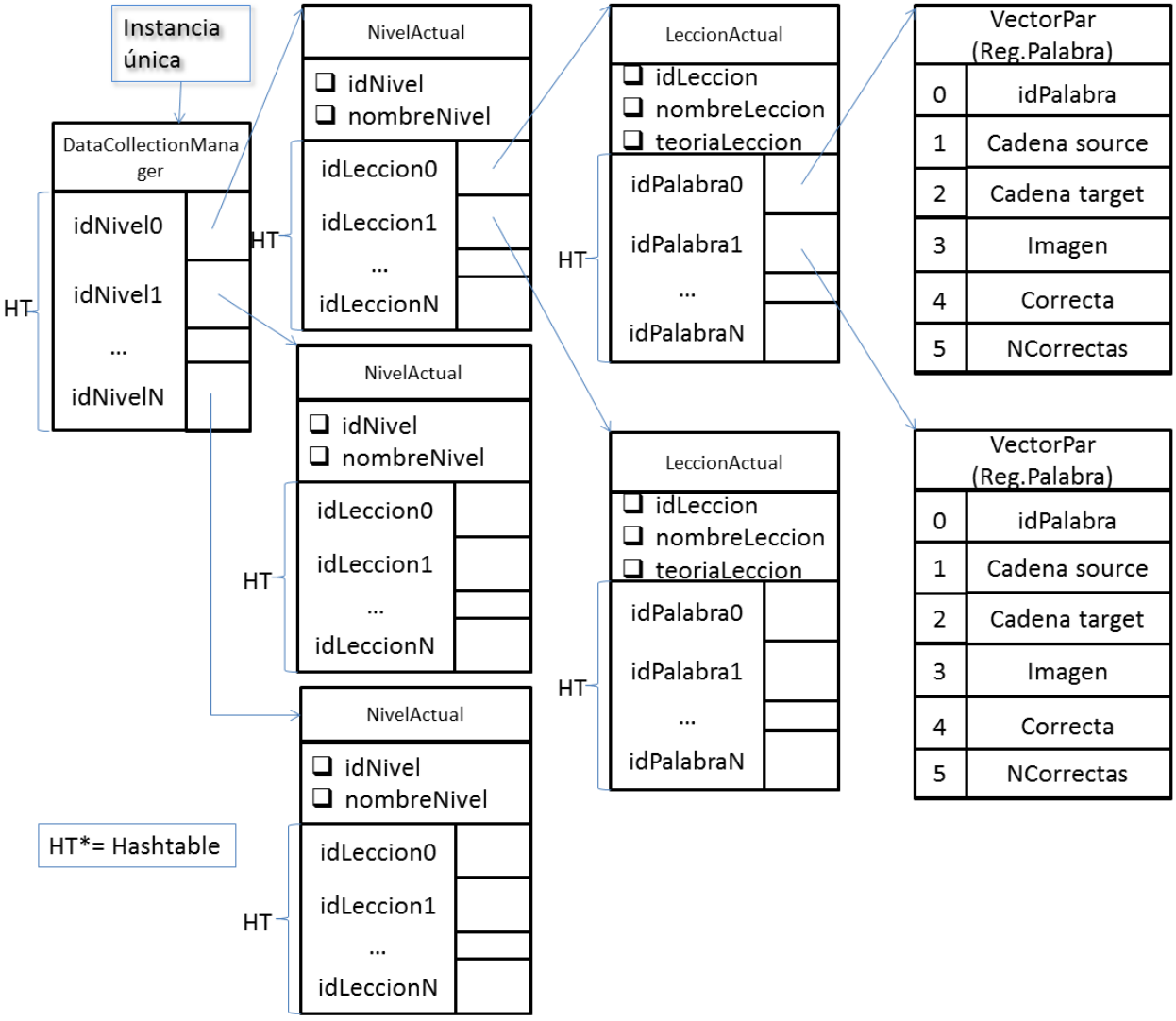


Figura 14: Estructura de Data Collection Manager ordenado por niveles

En OpsSobreDataCollectionManager disponemos de métodos para el intercambio entre datos desde BBDD y nuestro DataCollectionManager, como se introduce anteriormente, haciendo efectivo la técnica mapeo objeto-relacional (ORM).

9.2.3 VISTAS

Para el apartado de la implementación de las vistas representantes de la interfaz explicada anteriormente (4.2.3 Diseño de la interfaz gráfica de usuario), hemos optado por partir desde una vista principal común (Main), que sencillamente divide el flujo en vista para el alumno o vista para el profesor.

- Vista para el alumno

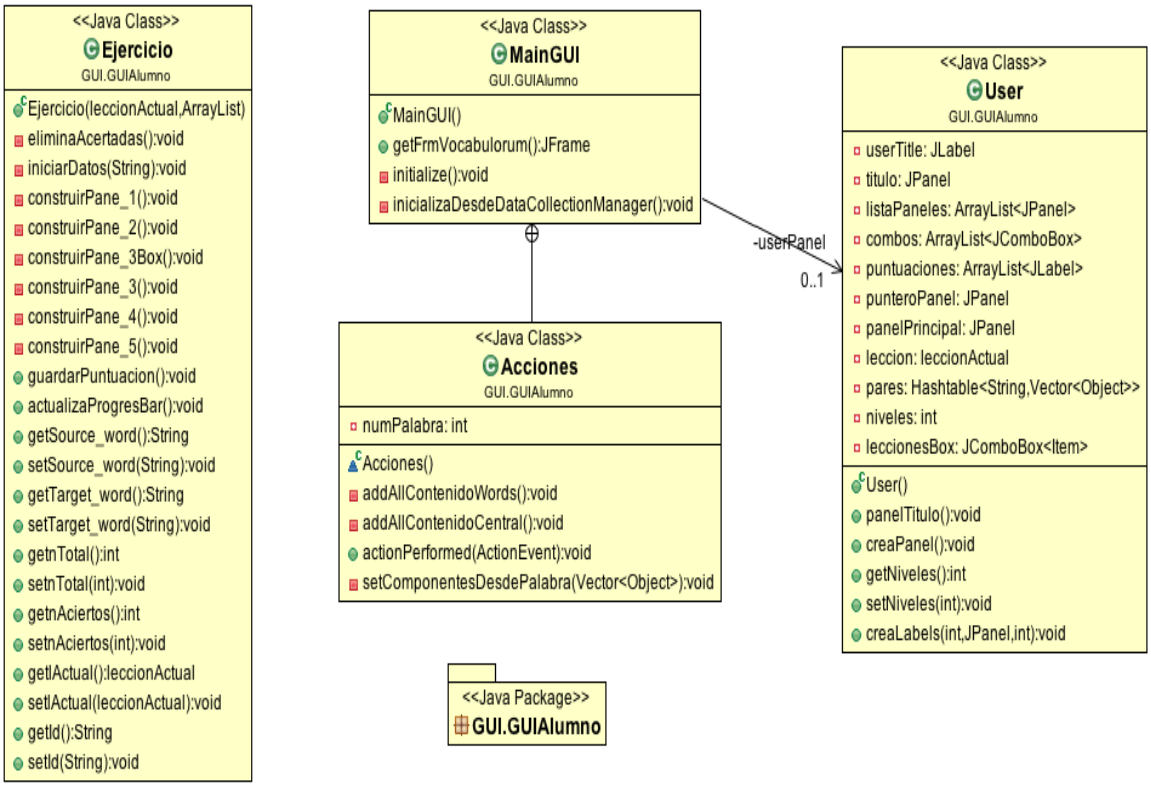


Figura 15: Diagrama de clases para vista Alumno.

En esta vista, definimos una clase de partida llamada MainGUIAlumno y a partir de los paneles que esta proporciona, invocamos otros paneles de las clases Ejercicio o User, y usamos la clase Acciones para controlar la vista, por tanto esta Clase pertenece la capa controlador.

- Vista para el profesor

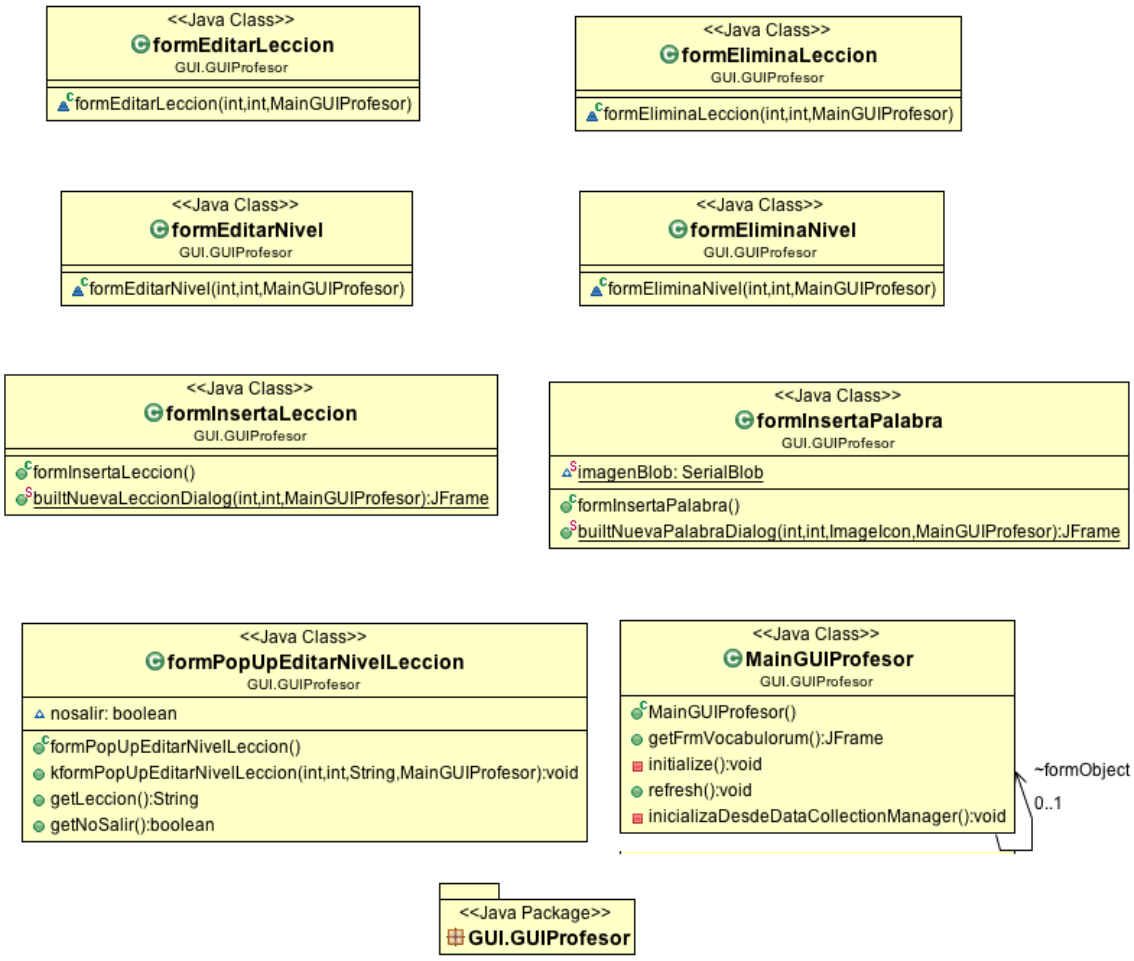


Figura 16: Diagrama de clases para vista Profesor.

Para la vista del módulo del profesor, partimos desde una ventana principal llamada MainGUIProfesor. Esta nos proporciona una tabla, para presentar todas las tuplas de palabras y todos los datos de estas (lección a la que pertenece, nivel, imagen, etc), y además permite editarlos.

También proporciona un menú para operar sobre el curso, con funciones para crear, abrir/guardar, importar/exportar, e insertar/modificar/eliminar aspectos propios de este. Para el primer grupo se emplean diálogos ya proporcionados por java como JFileChooser, y para el segundo grupo hemos realizado formularios a medida. Estos son uno para cada operación (forminsertaLeccion, forEliminaLeccion, formInsertaNivel, etc).

También este menú nos proporciona herramientas para operar sobre los datos como copiar, pegar, etc. y otras para la configuración inicial del programa.

9.2.4 CONTROLADORES

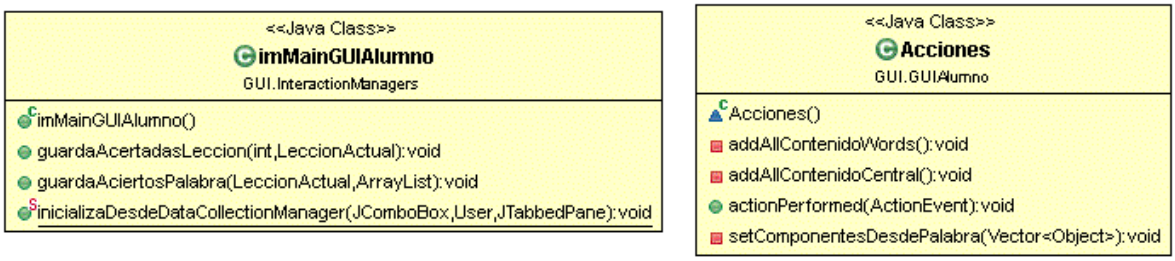
Las clases situadas entre el modelo y la vista, que hacen la función de controlador en nuestro proyecto son imMainGUIProfesor, para procesos de controlador del modulo del profesor y mainGUI.Acciones junto con imMainGUIAlumno para las del modulo del alumno.

- imMainGUIProfesor tiene métodos para cargar la vista con el objeto del curso (DataCollectionManager) residente en memoria, rellenar este objeto del curso residente en memoria desde la base de datos (explicado en 4.5.2 Modelo de datos, controlar las ediciones en la tabla y hacerlas efectivas en el curso, y mantener la puntuación del alumno actualizada.

- imMainGUIAlumno tiene métodos para ejercer modificaciones sobre el objeto del curso (DataCollectionManager). En nuestro caso solo son temas de puntuaciones, ya que el alumno no puede modificar nada más de un curso, solo escribir sus progresos.

- mainGUI.Acciones se encarga de ir haciendo visibles, o no visibles y habilitados o no habilitados los diferentes paneles de vistas o los botones y demás componentes gráficos según se va interactuando con la aplicación y además rellenando los componentes gráficos con datos del objeto curso según sea necesario.

Alumno:



Profesor:

Figura 17: Diagrama de clases para los controladores

9.2.5 OTRAS LIBRERÍAS USADAS

Además de las librerías nativas proporcionadas por java 7 hemos usado otras librerías externas.

9.2.5.1 SINTETIZADOR DE VOZ: FREETTS

FreeTTS es un motor de síntesis de voz de código libre para Java desarrollado por Sun Microsystems, basado en los sistemas de síntesis de voz Flite, Festival y FestVox desarrollados por la Universidad Carnegie Mellon y la Universidad de Edimburgo. Consta de un núcleo con el sistema de síntesis de voz, soporte de voces tanto propias como importadas a través de FestVox, CMU ARCTIC y MBROLA, y está implementado sobre librería original de síntesis de voz creada por Oracle JSAPI (The Java Speech API).

Posee un conjunto de interfaces que pueden ser usadas tanto como para sintetizar voces, para ser usado como servidor remoto back-end de aplicaciones de texto a voz en sistemas de telefonía móviles y evitarles el procesamiento, como para hacer aplicaciones "text-to-speech" de aplicaciones de escritorio o web.

Dado que está implementado sobre JSAPI, a nivel de implementación de código FreeTTS hace uso de los métodos y clases de la librería original de Oracle reusando muchos de ellos. A continuación explicaremos la organización general de ambas API haciendo hincapié en lo utilizado en el sintetizador usado en la práctica para reproducir las frases de vocabulario:

- Java Speech API
  - Las aplicaciones de voz sobre JSAPI no interactúan directamente con el hardware de audio de la maquina sobre la que se ejecutan. En vez de ello, hay una capa intermedia con el motor de Voz, que provee del uso de voz a la aplicación a modo de interfaz.

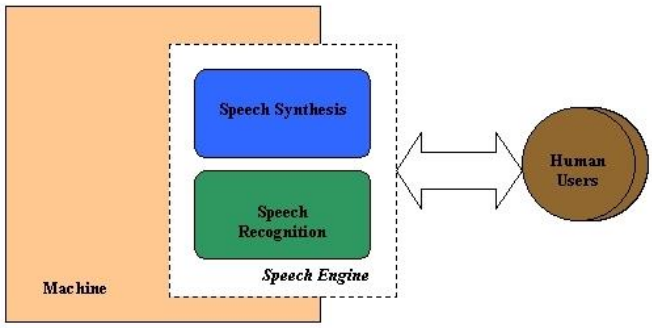


Figura 18: Motor de voz

  - JSAPI habilita la comunicación de las aplicaciones de voz construidas sobre su interfaz con los motores de voz de una manera común, estandarizada e independiente de implementación, de tal forma que se pueden usar motores de voz de diferentes desarrolladores a través de JSAPI mientras usen su estándar. Además, las aplicaciones podrán usar todas sus funcionalidades tales como usar diferentes idiomas de voz o el reconocimiento de habla.

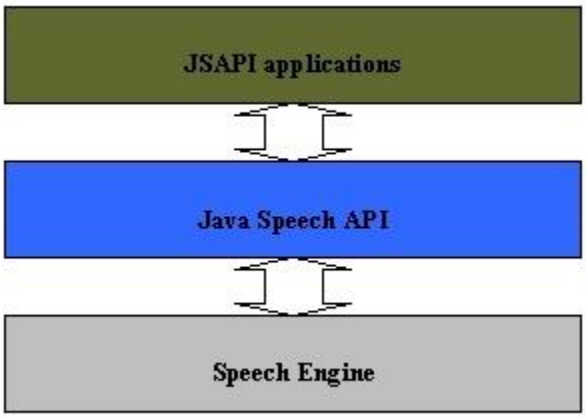


Figura 19: Pila de Java Speech API

- Estructura de clases e interfaces de Java Speech API:
  - javax.speech: Clases e interfaces para motores de voz genéricos.
  - javax.speech.synthesis Clases e interfaces para la síntesis de voz.
  - javax.speech.recognition Clases e interfaces para reconocimiento de voz.

Dado que no hemos usado el reconocimiento de voz, explicaremos el funcionamiento de los dos primeros paquetes de JSAPI:

- Paquete javax.speech:

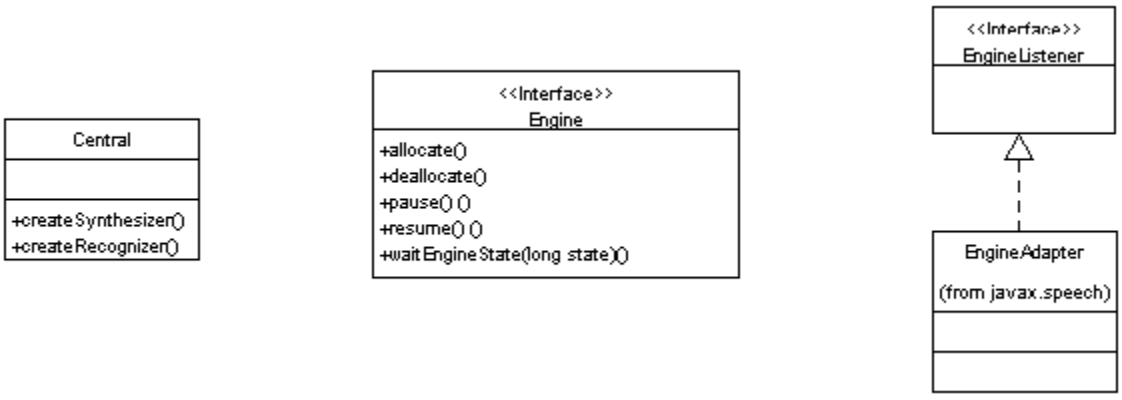


Figura 20: Diagrama de clases para javax.speech.

- CENTRAL

Clase factoría que usan todas las aplicaciones de JSAPI. Provee métodos estáticos para habilitar el acceso a los motores de síntesis y reconocimiento de voz.

- ENGINE

Interfaz que encapsula las operaciones genéricas que un motor de voz basado en JSAPI debe proveer a las aplicaciones de voz, tales como acciones para informar de las propiedades y el estado del motor de voz, reserve y liberación de recursos del motor de voz, así como mecanismos para pausar o continuar el sonido del flujo de audio generado o procesador por el motor de voz. Está subdividida en las interfaces Synthesizer y Recognizer, que definen funcionalidad añadida de sintetización y reconocimiento de voz, respectivamente.

JSAPI ha sido modelado para el procesamiento de eventos de componentes AWT, por lo que los eventos generados por el motor de voz pueden ser identificados y procesados, y para ello se usará la interfaz EngineListener o la clase EngineAdapter.

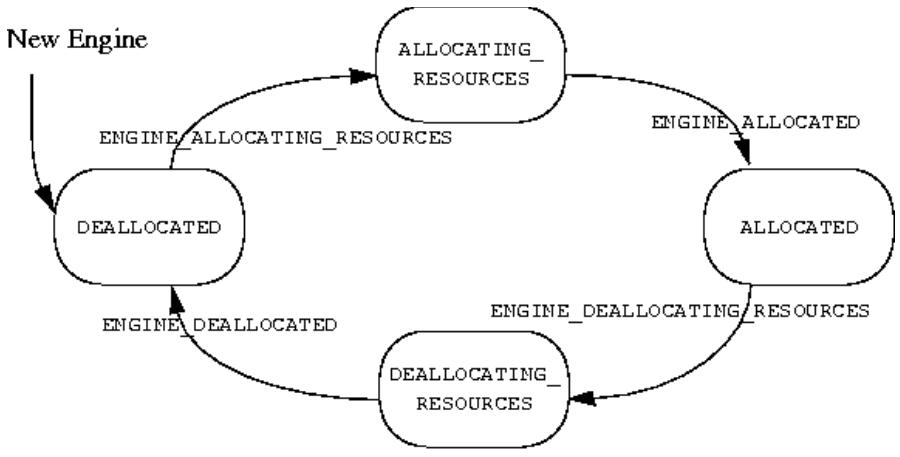


Figura 21: Diagrama de flujo de reserva de memoria

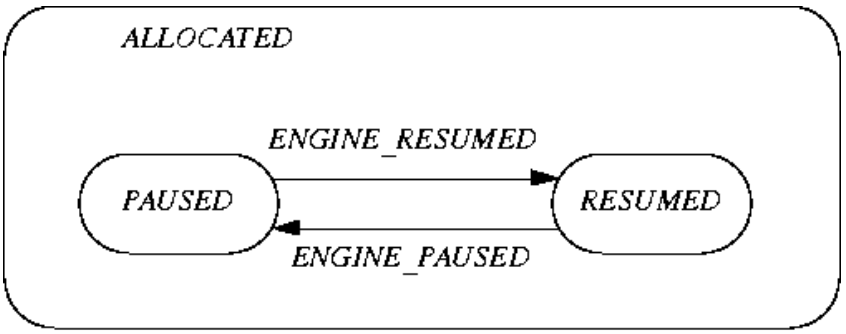


Figura 22: Diagrama de estados de pausa y reinicio de Engine

- Paquete javax.speech.synthesis:

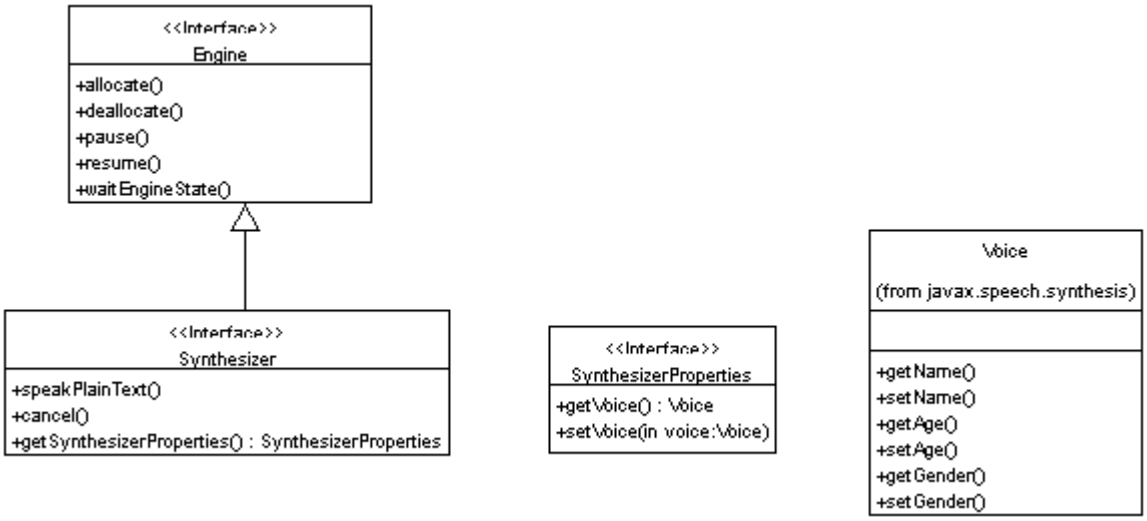


Figura 23: Diagrama de clases para javax.speech.synthesis

- SYNTHESIZER

La interfaz Synthesizer encapsula las operaciones que cualquier motor de sintetización de voz basado en JSAPI provee a las aplicaciones de voz, que esencialmente serán producir salida de voz desde una entrada de texto, detener el procesamiento de síntesis y otras operaciones relacionadas desde la interfaz Engine. La entrada de texto soportada es muy variada, desde Strings planos, URL hasta el lenguaje de marcado de propósito especial llamado Java Speech Markup Language (JSML).

- SynthesizerProperties

La interfaz SynthesizerProperties ofrece operaciones usadas para definir las propiedades en tiempo de ejecución para los objetos del tipo Synthesizer, incluyendo la voz, el volumen y el tono para la síntesis de la voz.

- Voice

La clase Voice representa la voz que el objeto Synthesizer usará para reproducir la salida de. Proporciona métodos para obtener metadatos de la voz usada para sintetizar por el objeto Synthesizer, tales como el nombre, edad y el género de la voz. Al igual que la interfaz Engine, los eventos generados durante la síntesis de voz pueden ser identificados y procesados o implementando los métodos de la interfaz SpeakableListener o usando la clase SpeakableAdapter.

❖ FreeTTS API

- Estructura de clases de FreeTTS:

- **com.sun.speech.engine** contiene el soporte para JSAPI 1.0. Incluye varios subclases, de las que detallaremos más adelante las que hemos usado
- **com.sun.speech.freetts** contiene la implementación del motor de síntesis de FreeTTS, y la mayoría del código de la interfaz. El paquete **com.sun.speech.freetts.jsapi** provee el nexo y la implementación de código de FreeTTS sobre JSAPI.

El paquete **com.sun.speech.freetts** se subdivide en las siguientes unidades:

- **com.sun.speech.freetts:** interfaces de alto nivel y clases de FreeTTS.
- **com.sun.speech.freetts.diphone:** soporte para voces codificadas difónicas.
- **com.sun.speech.freetts.clunits:** soporte para voces codificadas con unidades de cluster.
- **com.sun.speech.freetts.lexicon:** Implementación de Lexicons y reglas de letra a sonido.
- com.sun.speech.freetts.util: Herramientas y utilidades.

- com.sun.speech.freetts.audio: Soporte de salida de audio.
- **com.sun.speech.freetts.cart:** Implementación y soporte para árboles de regresión y clasificación (CART).
- **com.sun.speech.freetts.relp:** Soporte para decodificación Predictiva Lineal de Excitación Residual de ejemplos de (RELP).
- **com.sun.speech.freetts.en:** Contiene código específico del idioma inglés.
- **com.sun.speech.freetts.en.us:** Código específico del inglés americano.

Para realizar la síntesis y ejecución de voz, hay varios objetos relevantes que trabajan conjuntamente para llevarse a cabo. Detallaremos los necesarios para llevar a cabo la salida de voz desde texto en la aplicación:

- FreeTTSSpeakable

Interfaz que convertirá cualquier fuente de texto que necesite ser reproducida en un objeto del tipo FreeTTSSpeakable, su implementación por defecto es FreeTTSSpeakableImpl. Convierte la entrada de texto al formato que Synthesizer de JSAPI podrá leer (String, InputStream, JSML, XML) como FreeTTSSpeakable. Este objeto de le pasara a un objeto para ser reproducido.

- Voice

La clase Voice es el punto de proceso central en FreeTTS. Voice recibirá una entrada del tipo FreeTTSSpeakable, y traducirá su texto asociado en una voz y generará una salida de audio correspondiente a esa voz. Es la clase principal de personalización y modificación de sonidos y voces en FreeTTS, pudiendo configurar el idioma, el hablante y el algoritmo de personalización extendiendo desde un objeto de la clase. Aceptará el objeto FreeTTSSpeakable mediante el método Voice.speak y lo procesará en tres pasos:

- Voice convierte el objeto FreeTTSSpeakable en una serie de bloques de habla que dependerán del idioma utilizado, ya sea en delimitadores de texto, espacios o signos del idioma en particular.
- Mientras el objeto Voice genera cada bloque de habla, una serie de procesadores de bloques de habla los analizaran. Cada objeto Voice definirá su grupo de

procesadores de bloques de habla, y de esta forma se personaliza el resultado obtenido. Para cambiar como se juntan los bloques a la hora de realizar la síntesis de voz, solo basta con modificar el procesador de bloque de habla que contenga un nuevo algoritmo. Los procesadores trabajan a turno, anotando y modificando los bloques de habla con información, como la inserción de información de comienzo de frase o entonación.

- Una vez todos los procesadores de bloques de habla han procesado un bloque, el objeto Voice lo envía al procesador de salida de audio AudioOutput. Este procesador podrá correr en hilos separados para permitir que el procesamiento de bloques se solape con la salida de audio, asegurándonos la mínima latencia de audio posible. Es muy importante hacer buen uso de la reserva y liberación de memoria en la aplicación con los comandos proporcionados por JSAPI Y FreeTTS para su correcto funcionamiento con interfaces visuales y con listas de voces a reproducir amplias como en tablas de ejercicios.

- VoiceManager

VoiceManager es el repositorio principal de voces disponibles en FreeTTS. Para obtener la lista de voces por defecto, se podrá hacer:

```
VoiceManager voiceManager = VoiceManager.getInstance();

// crea una lista nueva de unidades de Voz

Voice[] voces = voiceManager.getVoices();

// recorrer la lista hasta encontrar una con las propiedad deseadas

...

// reservar recursos para la

voces[x].allocate();
```

Si se sabe la voz que se desea utilizar, se podrá obtener con el método voiceManager.getVoice(). Las voces podrán ser importadas desde librerías externas usando el soporte de FestVox, MBrola y similares.

9.2.5.2 LOOKANDFEEL: LIPSTIK

Web: http://lipstiklf.sourceforge.net/

El término "look and feel" (con el significado de "aspecto y tacto")[4] es una metáfora utilizada dentro del entorno de marketing para poder dar una imagen única a los productos, incluyendo áreas como el diseño exterior, trade dress, la caja en que se entrega al cliente, etc. incluyendo también la descripción de las características principales antes de su aparición (anuncio), durante la campaña de lanzamiento, y más tarde durante la posterior comercialización, con el fin de que el usuario, mediante todos estos parámetros, pueda llegar a conocer el producto y asociarlo con la marca de una forma inequívoca.

A nivel de API también puede tener un determinado aspecto y comportamiento. Diferentes partes de una API (por ejemplo, algunas clases o paquetes) a menudo están vinculadas por convenciones sintácticas y semánticas (por ejemplo, por el modelo asincrónico ejecución, o por la forma accede a los atributos de un objeto). Estos elementos se representan de forma explícita (es decir, son parte de la sintaxis de la API) o implícitamente (es decir, son parte de la semántica de la API).

La arquitectura Swing de java, ha sido diseñada para permitirnos cambiar el “look and feel” (aspecto y sensación) de nuestras aplicaciones java. La función fue introducida a partir de java 4. Existen diferentes librerías para poder aplicar esta función, algunas internas a java (metal seria la estándar de java, nimbus una librería que empezó externa pero fue adoptada en las últimas versiones de java, y librerías especificas para cada plataforma, aéreo de Windows vista, o quaqua para mac o __ para linux), y otras externas como la usada en el proyecto, lipStick.

Sobre LipStikLookAndFeel es un proyecto que fue registrado en SourceForge.net en Noviembre de 2006 y pretende ser un componente similar al fabuloso tema de KDE* llamado LIPstilk.

9.2.6 PATRONES DE DISEÑO

Una cosa que los diseñadores expertos no hacen es resolver cada problema desde el principio. A menudo reutilizan las soluciones que ellos han obtenido en el pasado. Cuando encuentran una buena solución, utilizan esta solución una y otra vez. Consecuentemente tú podrías encontrar patrones de clases y comunicaciones entre objetos en muchos sistemas orientados a objetos. Estos patrones solucionan problemas específicos del diseño y hacen los diseños orientados a objetos más flexibles, elegantes y por último reutilizables. Los patrones de diseño ayudan a los diseñadores a reutilizar con éxito diseños para obtener nuevos diseños. Un diseñador que conoce algunos patrones puede aplicarlos inmediatamente a problemas de diseño sin tener que descubrirlos.

El valor de la experiencia en los diseños es muy importante. Cuando tienes que hacer un diseño que crees que ya has solucionado antes pero no sabes exactamente cuando o como lo hiciste. Si pudieras recordar los detalles de los problemas anteriores y como los solucionaste, entonces podrías reutilizar la experiencia en vez de tener que volver a pensarlo.

El objetivo de los patrones de diseño es guardar la experiencia en diseños de programas orientados a objetos. Cada patrón de diseño nombra, explica y evalúa un importante diseño en los sistemas orientados a objetos. Es decir se trata de agrupar la experiencia en diseño de una forma que la gente pueda utilizarlos con efectividad. Por eso se han documentado los más importantes patrones de diseño y presentado en catálogos.

Los patrones de diseño hacen más fácil reutilizar con éxito los diseños y arquitecturas. Expresando estas técnicas verificadas como patrones de diseño se hacen más accesibles para los diseñadores de nuevos sistemas. Los patrones de diseño te ayudan a elegir diseños alternativos que hacen un sistema reutilizable y evitan alternativas que comprometan la reutilización. Los patrones de diseño pueden incluso mejorar la documentación y mantenimiento de sistemas existentes. Es decir, los patrones de diseño ayudan a un diseñador a conseguir un diseño correcto rápidamente.

9.2.6.1 SINGLETON [GOF95]

Garantiza que solamente se crea una instancia de la clase y provee un punto de acceso global a él. Todos los objetos que utilizan una instancia de esa clase usan la

misma instancia. Respecto a la aplicabilidad el patrón Singleton se puede utilizar en los siguientes casos:

- Debe haber exactamente una instancia de la clase.
- La única instancia de la clase debe ser accesible para todos los clientes de esa clase.

El patrón Singleton es relativamente simple, ya que sólo involucra una clase. Una clase Singleton tiene una variable static que se refiere a la única instancia de la clase que quieres usar. Esta instancia es creada cuando la clase es cargada en memoria. Tú deberías de implementar la clase de tal forma que se prevenga que otras clases puedan crear instancias adicionales de una clase Singleton. Esto significa que debes asegurarte de que todos los constructores de la clase son privados.

Para acceder a la única instancia de una clase Singleton, la clase proporciona un método static, normalmente llamado getInstancia, el cual retorna una referencia a la única instancia de la clase.

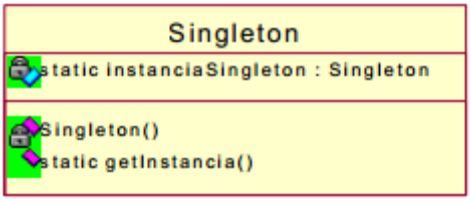


Figura 24: Clase Singleton

Nosotros hemos aplicado este patrón para hacer un objeto único de DataCollectionManager (objeto representante del curso). De manera que si no existe se crea la instancia única y se rellena con los datos de mapeo entidad-relación, y si ya existe se devuelve la instancia única, asegurando así que el único es único.

9.2.6.2 ITERATOR [GOF95]

Define un interface que declara métodos para acceder secuencialmente a los objetos de una colección. Una clase que accede a una colección solamente a través de un interface independiente de la clase que implementa el interface. Sobre su aplicabilidad el patrón Iterator se puede utilizar en los siguientes casos:

- Una clase necesita acceder al contenido de una colección sin llegar a ser dependiente de la clase que es utilizada para implementar la colección, es decir sin tener que exponer su representación interna.

- Una clase necesita un modo uniforme de acceder al contenido de varias colecciones.

- Cuando se necesita soportar múltiples recorridos de una colección.

El diagrama de clases que muestra la organización de las clases e interfaces que participan en el patrón Iterator es el siguiente:

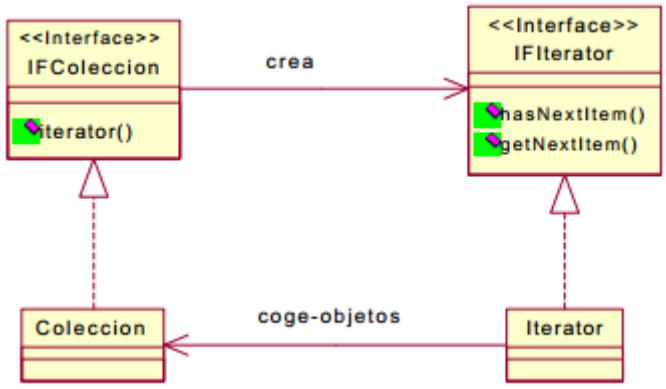


Figura 25: Patrón Iterator

A continuación están las descripciones de los papeles que juegan las clases e interfaces en la organización citada anteriormente:

- **Colección**: Una clase en este papel encapsula una colección de objetos o valores.

- **IFIterator**: Un interface en este papel define los métodos que acceden secuencialmente a los objetos que están encapsulados en un objeto *Coleccion*.

- **Iterator:** Una clase en este papel implementa un interface *IFIterator*. Sus instancias proporcionan acceso secuencial a los contenidos del objeto *Coleccion* asociado con el objeto *Iterator*.

- **IFColeccion:** Las clases *Coleccion* normalmente toman responsabilidad para crear sus propios objetos *Iterator*. Es conveniente tener un modo consistente de preguntar a un objeto *Coleccion* para crear un objeto *Iterator* por si mismo. Para proporcionar esta consistencia, todas las clases *Coleccion* implementan un interface *IFColeccion* que declara un método para crear objetos *Iterator*.

Respecto al uso que hemos dado a este patrón, se encuentra en la implementación de los objetos herederos de *hashtables* que usamos para representar Niveles o Lecciones. Estos deben poder ser recorridos para obtener datos globales a todas los niveles o a todas las lecciones o incluso al curso por completo, por eso necesitamos un patrón robusto para estos recorridos.

10 APÉNDICES

10.1 APÉNDICE 1: GUÍA DE USUARIO

La aplicación Hello word! es sencilla e intuitiva.

Al iniciar la aplicación nos encontramos con una imagen de presentación donde el usuario elegirá si va a iniciar la aplicación como alumno o como profesor. Con tan solo hacer un click en una de las dos imágenes se iniciará la aplicación con el perfil deseado.

Figura 1- Pantalla inicio

➢ Aplicación modo alumno

Una vez iniciada la aplicación en modo alumno, se podrá seleccionar, dentro de la pestaña Aprendizaje, el nivel al que se desea acceder y la lección correspondiente. Una vez seleccionados se hará click en el botón OK.

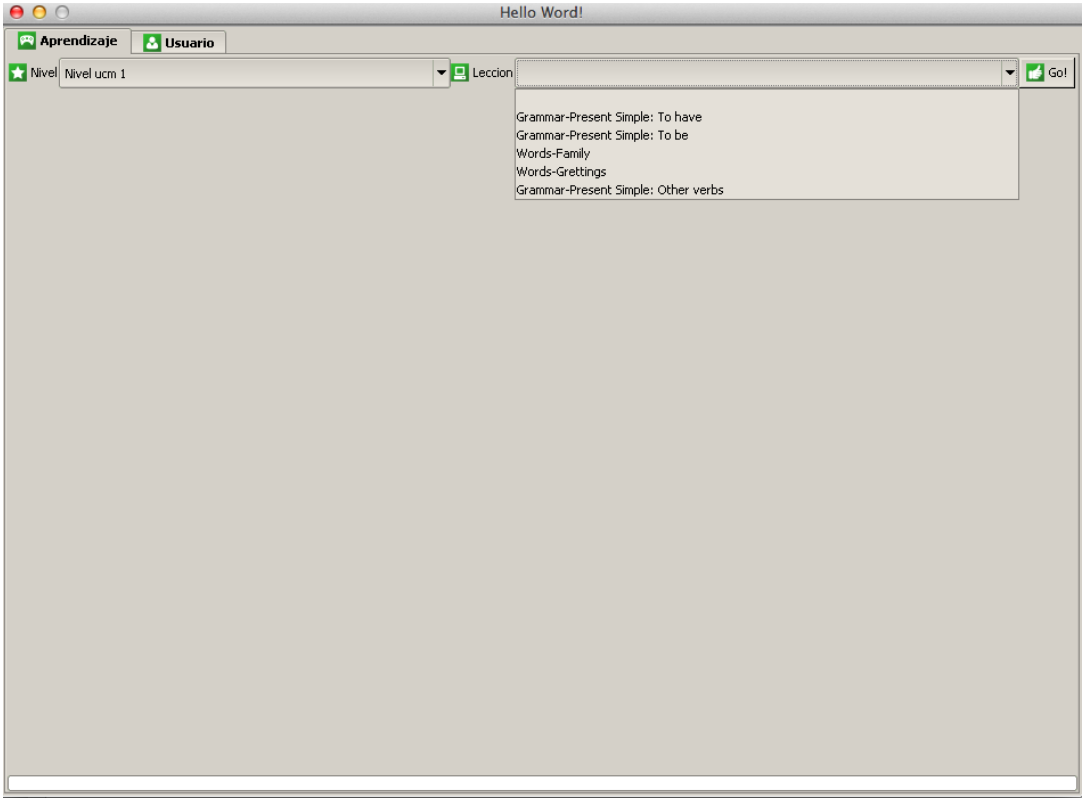


Figura 2- Selección nivel y lección

Una vez seleccionado el nivel y la lección se accederá al contenido de la lección, donde ser presentará una introducción a la lección y dos botones, uno para acceder a ver las palabras o frases de ejemplo de la lección y otro para realizar el ejercicio asociado a la lección.

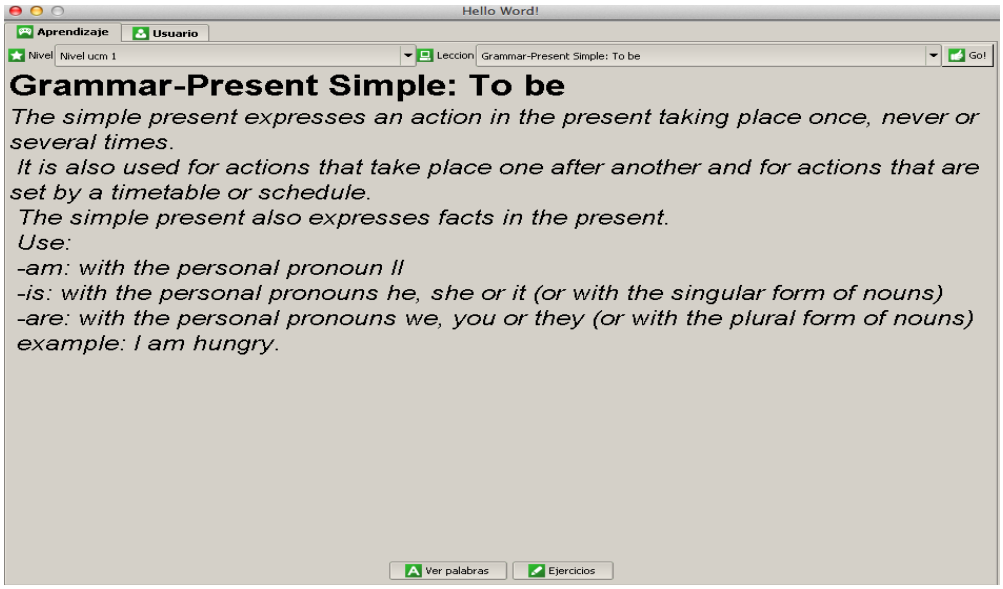


Figura 3- Introducción a la lección

Si el usuario selecciona el botón de "Ver palabras" mostraremos una vista donde puede consultarse palabras o frases en castellano con su correspondiente traducción en inglés. Además se podrá ver una imagen explicativa, en el caso de las palabras, y escuchar la correcta pronunciación con el archivo de sonido.

Figura 4- Navegación por palabras y frases

Una vez dentro de la vista de navegación por el contenido de la lección, el usuario puede hacer click en los distintos botones de navegación: inicio, fin, siguiente y anterior.

Si el usuario selecciona el botón de "Ejercicio" accederá a la vista donde comprobar sus conocimientos.

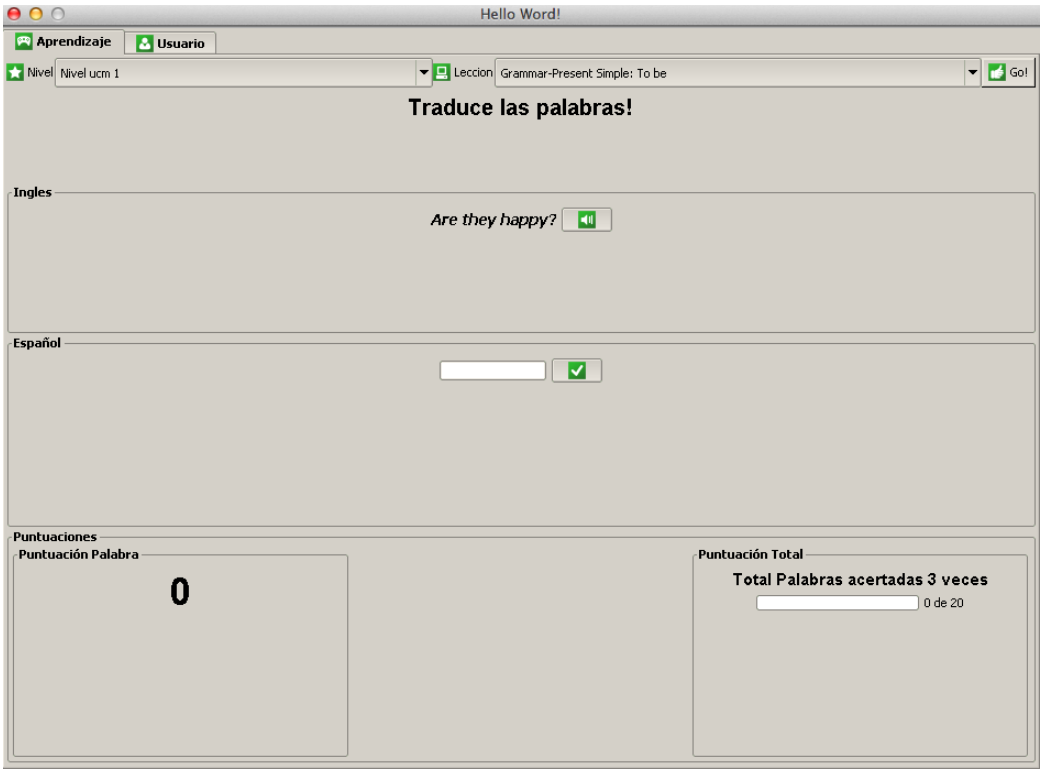


Figura 5- Vista inicio del ejercicio

El usuario tendrá que escribir la solución correcta en idioma castellano en el cuadro de texto y darle al botón de "Aceptar". Inmediatamente le aparecerá la respuesta correcta y un menú con una pregunta donde se le cuestiona si la respuesta escrita por el usuario coincide con la mostrada. Si el usuario ha respondido correctamente seleccionará "Si" y sino "No".

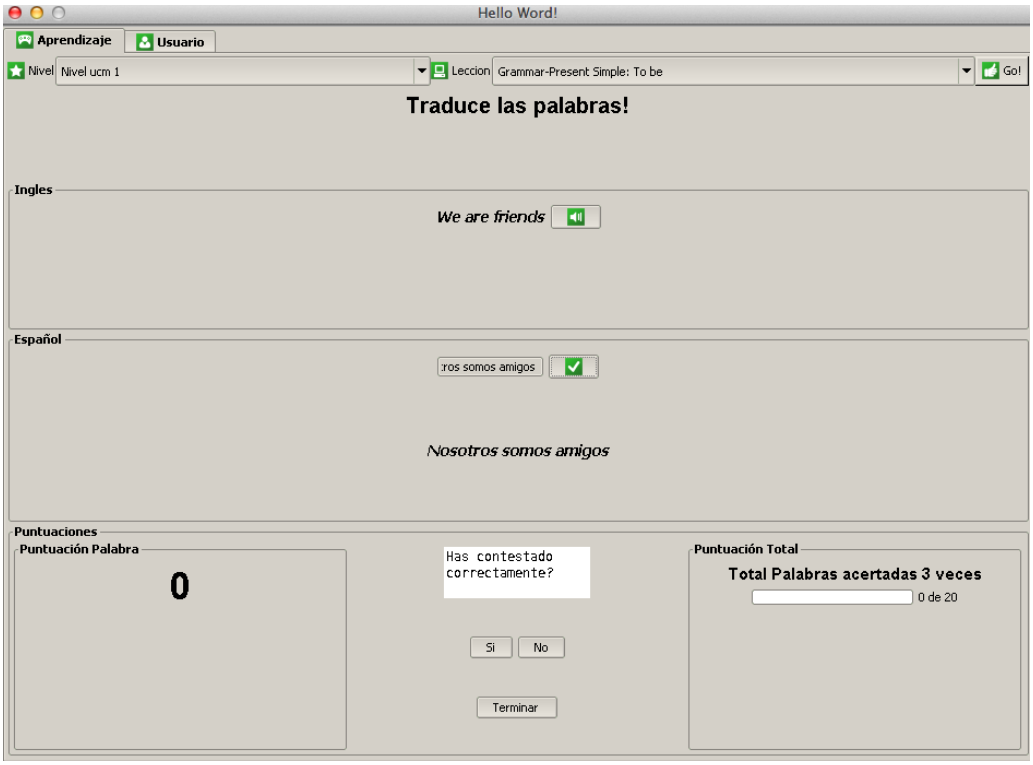


Figura 6- Vista del ejercicio con la respuesta

Además podemos ver que aparece la "puntuación de la palabra" que es el valor que corresponde con el número de veces acertada ésa palabra o frase, y la puntuación total, que irá indicando cuantas palabras o frases se han acertado tres veces.

El usuario también tiene disponible el botón "Terminar" para salir del ejercicio y guardar los resultados cuando lo deseé.

Por otro lado podemos ver la pestaña "Usuario", donde el usuario podrá consultar las puntuaciones por cada lección seleccionada en el menú desplegable de cada nivel.

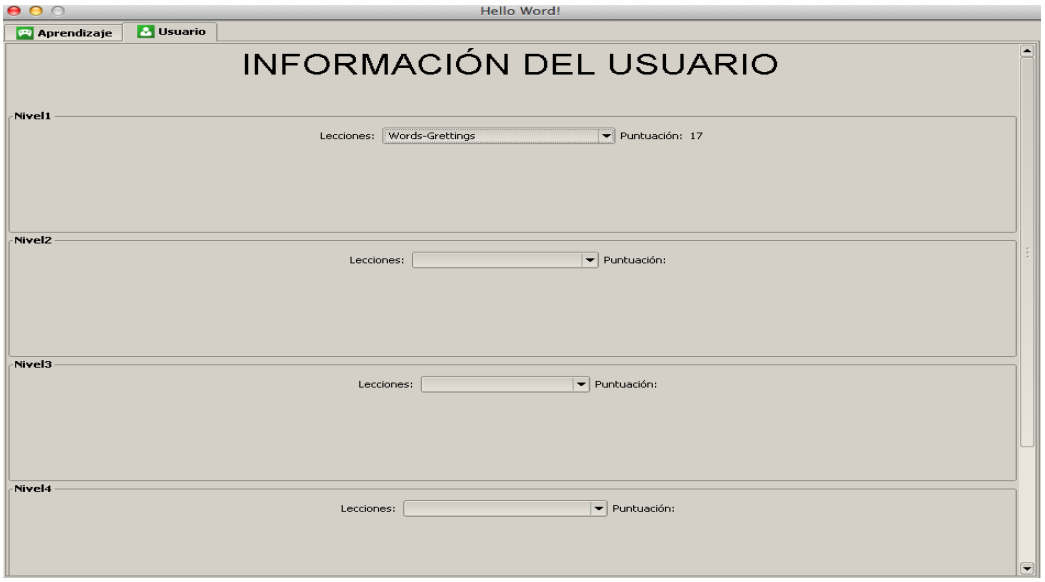


Figura 7- Información del usuario

Para que el usuario vaya siguiendo el método en el orden correcto, el usuario está obligado a aprobar cada lección antes de poder continuar a la siguiente.

Si en el menú correspondiente a seleccionar nivel y lección elige uno incorrecto le aparecerá un mensaje informativo indicándole que no puede continuar a la siguiente lección sin haber superado la actual.

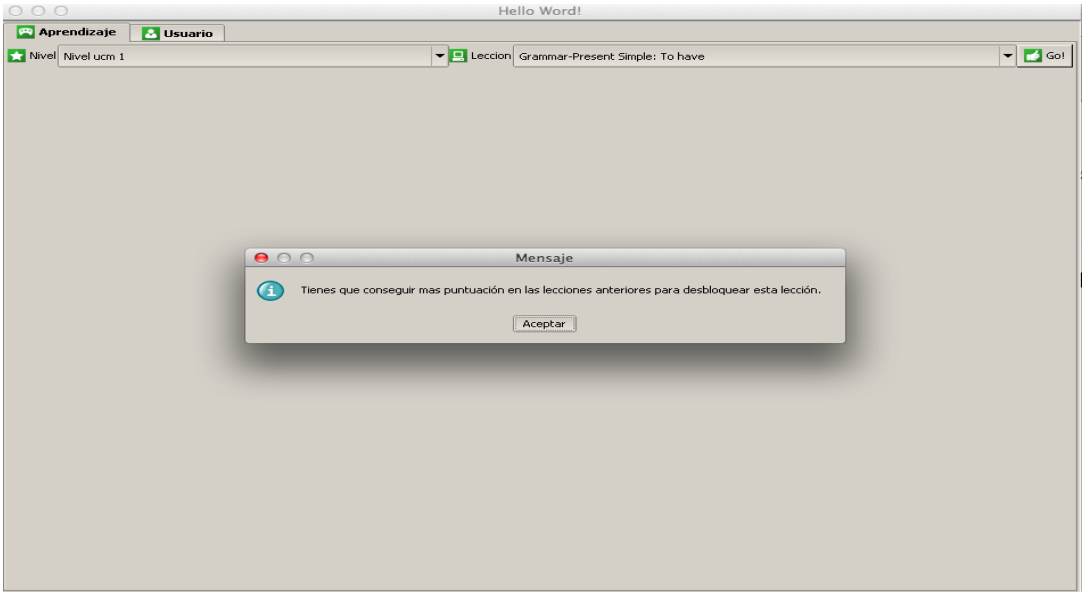


Figura 8- Mensaje de error al seleccionar nivel y lección

➢ Aplicación modo profesor

Cuando se inicia la aplicación en modo profesor, nos encontramos con una barra de herramientas y una tabla con distintas columnas en blanco. Éste es el panel de edición de la base de datos para el profesor. Desde éste menú el usuario tiene una serie de opciones, explicadas a continuación, para crear una base de datos o editar la base de datos por defecto.

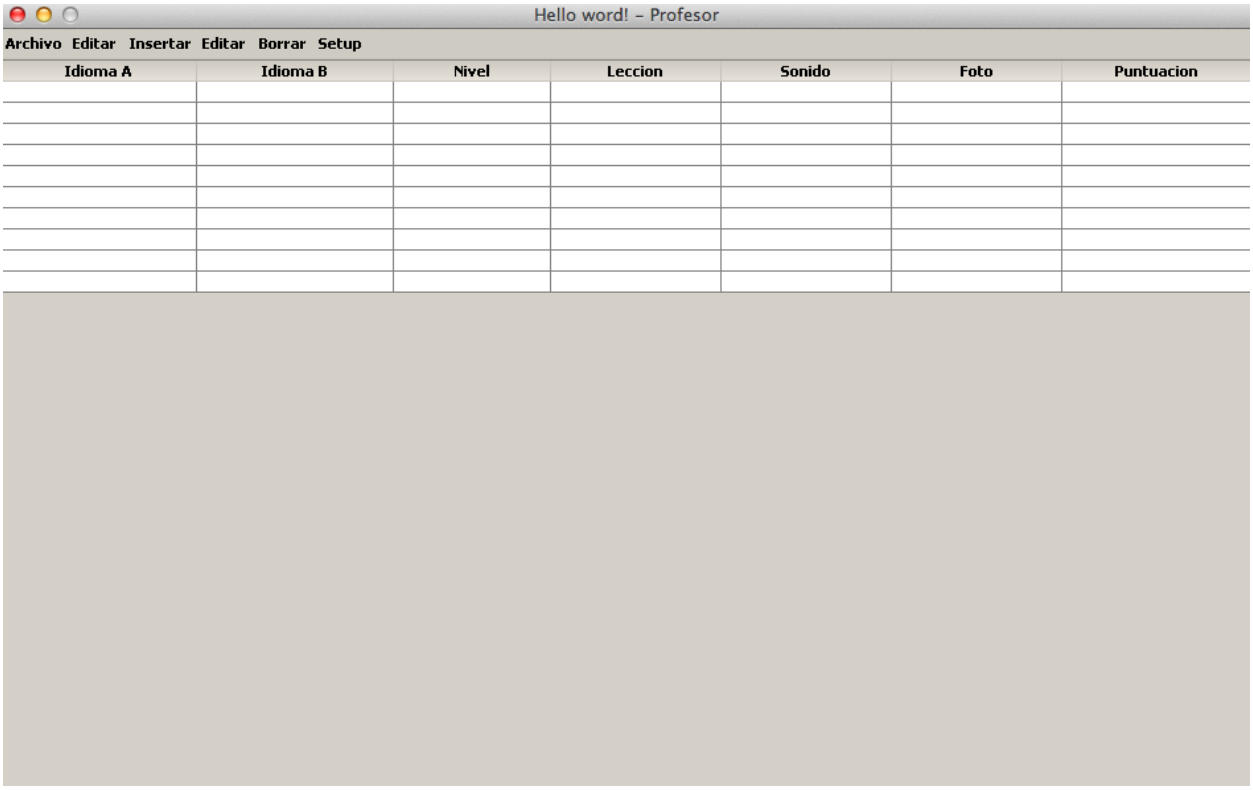


Figura 9- Pantalla inicio en modo profesor

Si el usuario hace click en la primera opción de la barra de herramientas, en "Archivo", se encontrará con las opciones:

- Nuevo: Opción para crear una base de datos nueva.
- Abrir: Opción para abrir una base de datos desde fichero o desde base de datos.
- Guardar: Opción para guardar a un fichero o a una base de datos.

- Importar: Opción para importar una base de datos.
- Exportar: Opción para exportar una base de datos.
- Salir: Opción para salir de la aplicación.

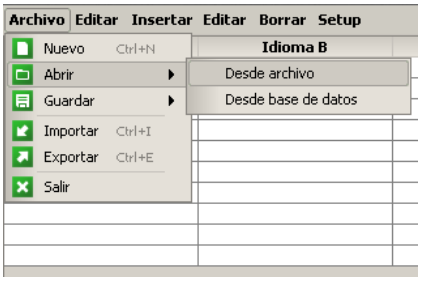


Figura 10- Opción Abrir

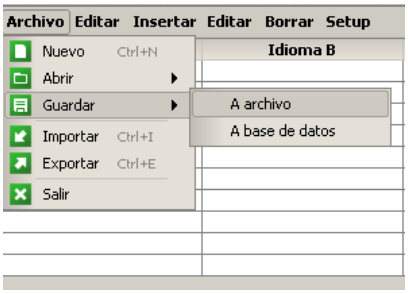
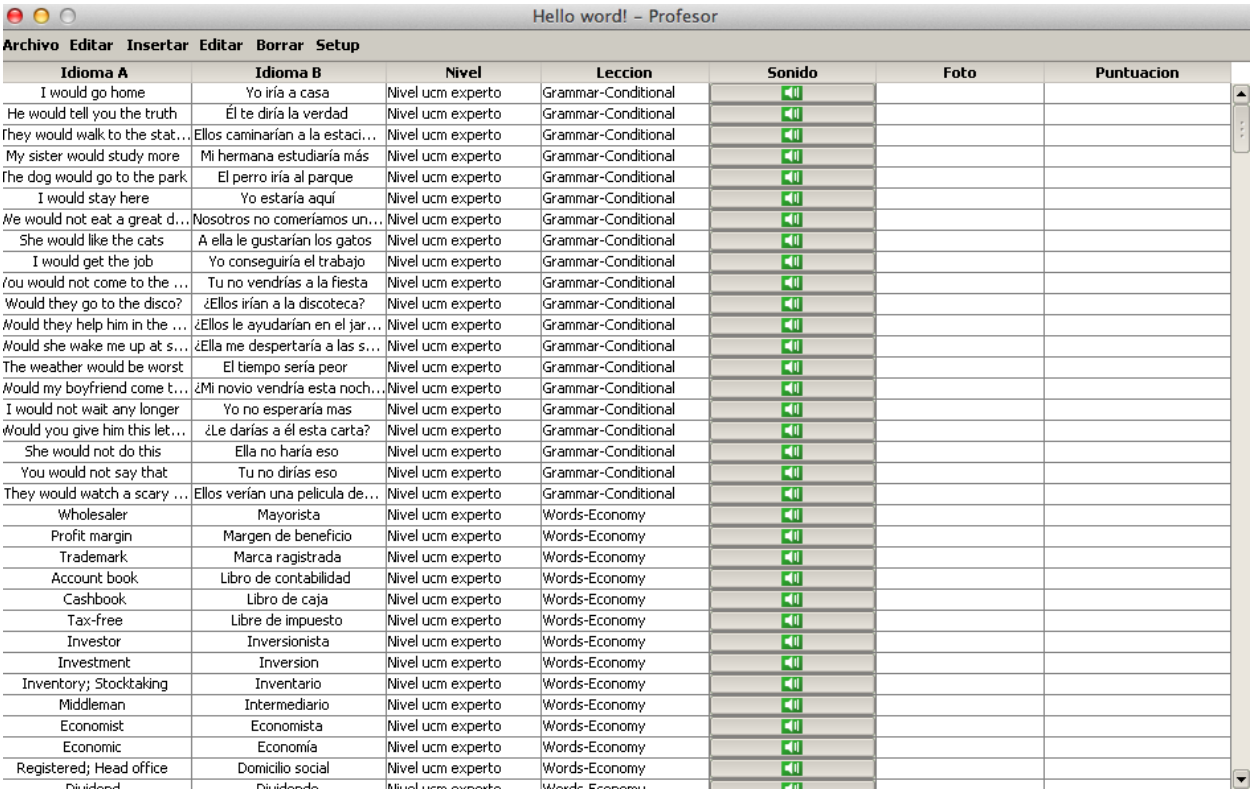


Figura 11- Opción Guardar.

Una vez el usuario tenga datos abiertos y preparados para editar la vista se mostraría como la siguiente:



Idioma A	Idioma B	Nivel	Leccion	Sonido	Foto	Puntuacion
I would go home	Yo iría a casa	Nivel ucm experto	Grammar-Conditional			
He would tell you the truth	Él te diría la verdad	Nivel ucm experto	Grammar-Conditional			
They would walk to the stat...	Ellos caminarían a la estaci...	Nivel ucm experto	Grammar-Conditional			
My sister would study more	Mi hermana estudiaría más	Nivel ucm experto	Grammar-Conditional			
The dog would go to the park	El perro iría al parque	Nivel ucm experto	Grammar-Conditional			
I would stay here	Yo estaría aquí	Nivel ucm experto	Grammar-Conditional			
We would not eat a great d...	Nosotros no comeríamos un...	Nivel ucm experto	Grammar-Conditional			
She would like the cats	A ella le gustarían los gatos	Nivel ucm experto	Grammar-Conditional			
I would get the job	Yo conseguiría el trabajo	Nivel ucm experto	Grammar-Conditional			
You would not come to the ...	Tu no vendrías a la fiesta	Nivel ucm experto	Grammar-Conditional			
Would they go to the disco?	¿Ellos irían a la discoteca?	Nivel ucm experto	Grammar-Conditional			
Would they help him in the ...	¿Ellos le ayudarían en el jar...	Nivel ucm experto	Grammar-Conditional			
Would she wake me up at s...	¿Ella me despertaría a las s...	Nivel ucm experto	Grammar-Conditional			
The weather would be worst	El tiempo sería peor	Nivel ucm experto	Grammar-Conditional			
Would my boyfriend come t...	¿Mi novio vendría esta noch...	Nivel ucm experto	Grammar-Conditional			
I would not wait any longer	Yo no esperaría mas	Nivel ucm experto	Grammar-Conditional			
Would you give him this let...	¿Le darías a él esta carta?	Nivel ucm experto	Grammar-Conditional			
She would not do this	Ella no haría eso	Nivel ucm experto	Grammar-Conditional			
You would not say that	Tu no dirías eso	Nivel ucm experto	Grammar-Conditional			
They would watch a scary ...	Ellos verían una pelicula de...	Nivel ucm experto	Grammar-Conditional			
Wholesaler	Mayorista	Nivel ucm experto	Words-Economy			
Profit margin	Margen de beneficio	Nivel ucm experto	Words-Economy			
Trademark	Marca ragistrada	Nivel ucm experto	Words-Economy			
Account book	Libro de contabilidad	Nivel ucm experto	Words-Economy			
Cashbook	Libro de caja	Nivel ucm experto	Words-Economy			
Tax-free	Libre de impuesto	Nivel ucm experto	Words-Economy			
Investor	Inversionista	Nivel ucm experto	Words-Economy			
Investment	Inversion	Nivel ucm experto	Words-Economy			
Inventory; Stocktaking	Inventario	Nivel ucm experto	Words-Economy			
Middleman	Intermediario	Nivel ucm experto	Words-Economy			
Economist	Economista	Nivel ucm experto	Words-Economy			
Economic	Economía	Nivel ucm experto	Words-Economy			
Registered; Head office	Domicilio social	Nivel ucm experto	Words-Economy			

Figura 12- Vista con datos cargados

Si el usuario selecciona la opción “Editar” dentro de la barra de herramientas, un menú le aparecerá con las opciones comunes de edición: Deshacer, Cortar, Copiar, Pegar, Eliminar, Seleccionar todo.

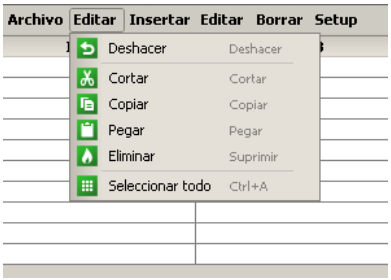


Figura 13- Opciones Edición

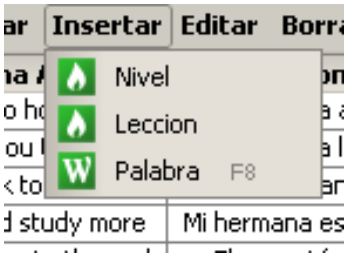


Figura 14- Opciones Insertar

Si el usuario selecciona la opción de “Insertar” de la barra de herramientas, podrá insertar un nivel, lección o palabra nueva en la base de datos.

Si selecciona insertar un nuevo nivel aparecerá una nueva ventana para insertar el nombre del nuevo nivel.

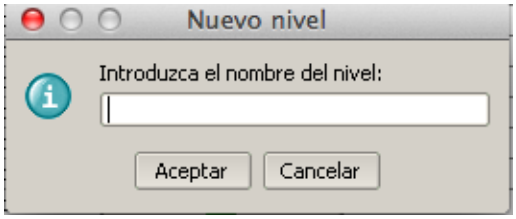


Figura 15- Insertar nivel

Si por el contrario, el usuario selecciona la opción para insertar una nueva lección, un menú nuevo aparecerá donde el usuario tendrá que indicar a qué nivel pertenecerá la nueva lección, que nombre tendrá y podrá escribir la parte de introducción teórica de la lección en un cuadro de texto.

Por último, el usuario puede elegir insertar una palabra. Aparecerá una nueva ventana donde tendrá que indicar a qué nivel y lección pertenecerá esta nueva palabra.

Después escribirá en los cuadros de texto la palabra en inglés y en castellano, y podrá insertar la imagen descriptiva si lo desea.

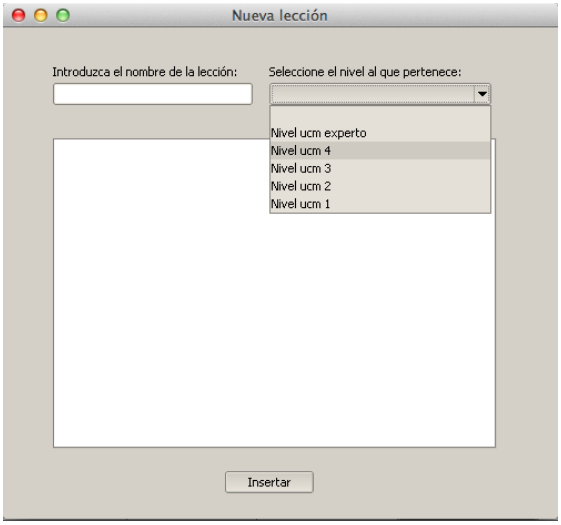


Figura 16- Insertar lección

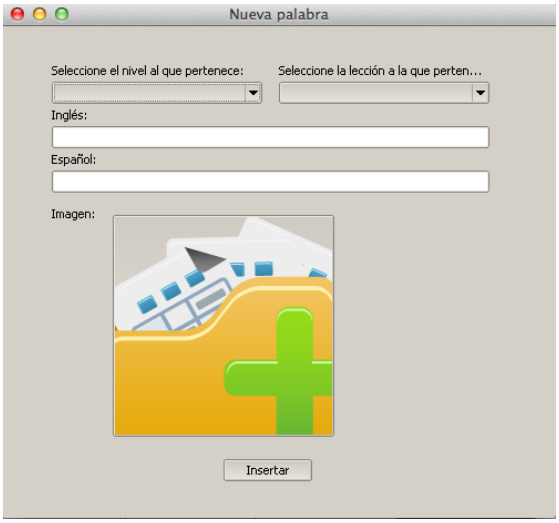


Figura 17- Insertar palabra

Si el usuario selecciona la opción "Editar" de la barra de herramientas podrá editar tanto un nivel como una lección.

Si selecciona editar un nivel, una nueva ventana aparecerá donde desde un menú desplegable podrá seleccionar el nivel a editar, y en el cuadro de texto editar el nombre del nivel.

Figura 18- Menú Editar

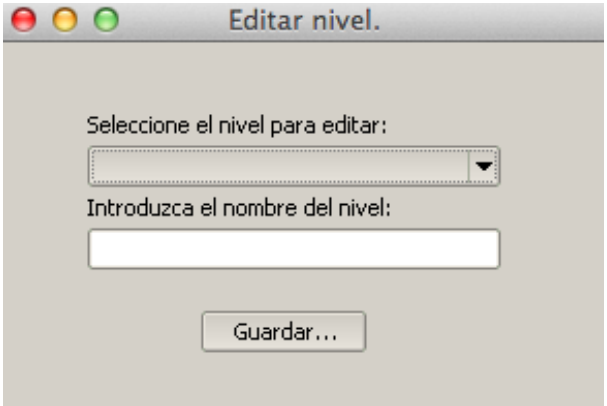


Figura 19- Editar nivel

Si el usuario selecciona la opción para editar la lección una nueva ventana aparecerá con un par de menús desplegables para que el usuario seleccione que lección quiere editar y a qué nivel pertenece. Además dos cuadros de texto aparecerán, uno para editar el título de la lección y otro para editar el contenido de introducción teórica de la lección.

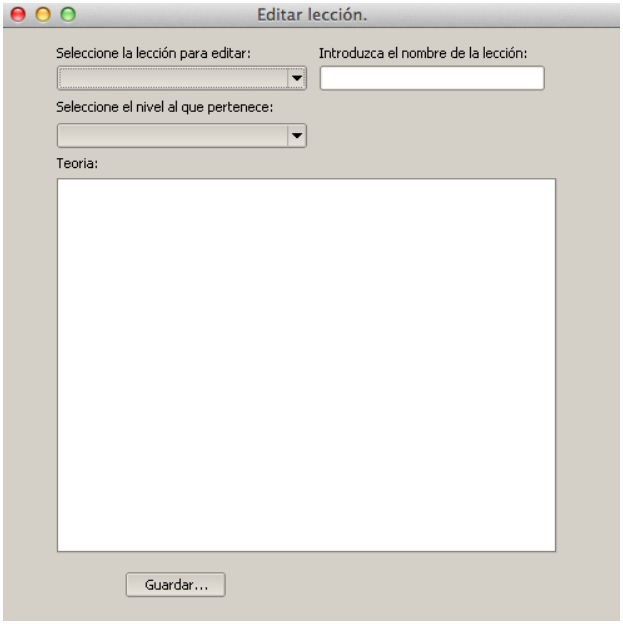


Figura 20- Editar lección

Si el usuario selecciona la opción "Borrar" de la barra de herramientas, podrá borrar un nivel o una lección de los datos.

Figura 21- Opción borrar

Si selecciona borrar un nivel una nueva ventana aparecerá con un menú desplegable para seleccionar el nivel que se desea borrar.

Si por el contrario selecciona eliminar una lección, una nueva ventana aparecerá con dos menús desplegables, uno para seleccionar el nivel al que pertenece la lección que se quiere eliminar, y otro para indicar qué lección es la que se desea eliminar.

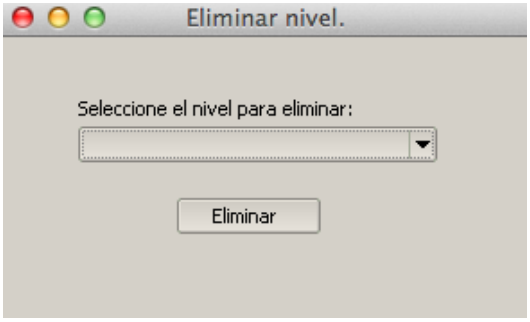


Figura 22- Borrar nivel

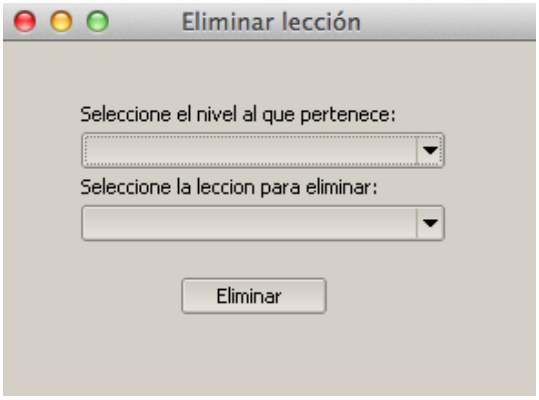


Figura 23- Borrar lección

Por último, si el usuario selecciona la opción "Setup" del menú de herramientas, podrá cargar la base de datos, por defecto en la aplicación.

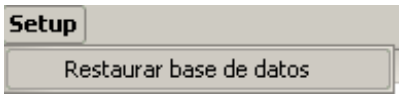


Figura 24- Opción Setup

10.2 Apéndice 2: Guía de instalación

10.2.1 Apéndice 2.1. Instalación de hello word!

Para ello debemos ejecutar archivo setup.exe, se abrirá el instalador y en primer lugar se comprobará si hay una versión instalada en el ordenador de java runtime machine superior a la versión 1.7.20. Si no esta instalada se procederá a instalarse.

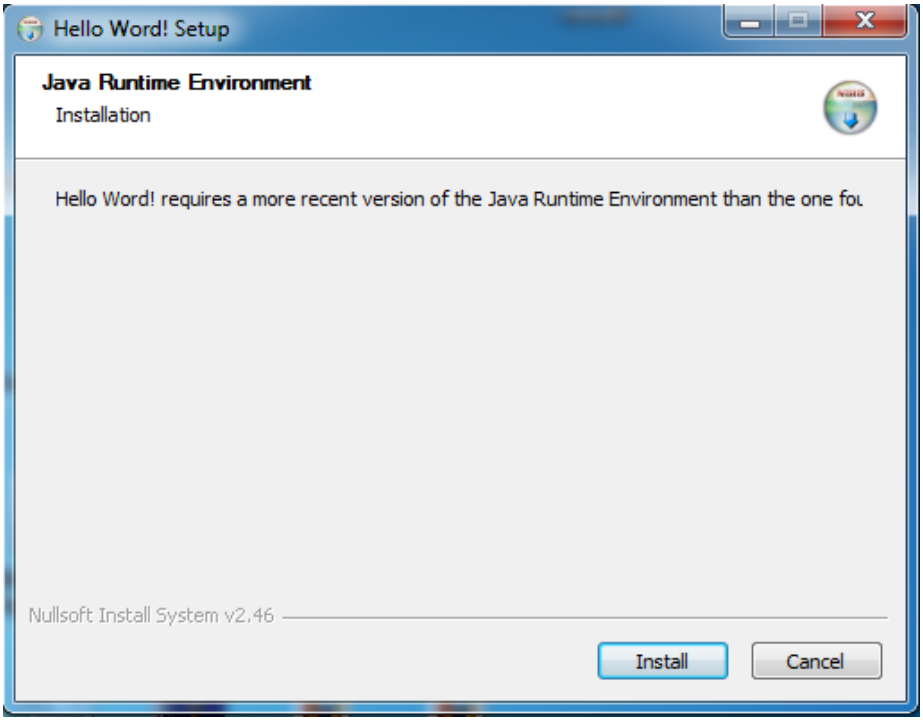


Figura 1- Pantalla 1 instalación JRE

Solo tendremos que pulsar "Install" y deberemos esperar varios minutos.

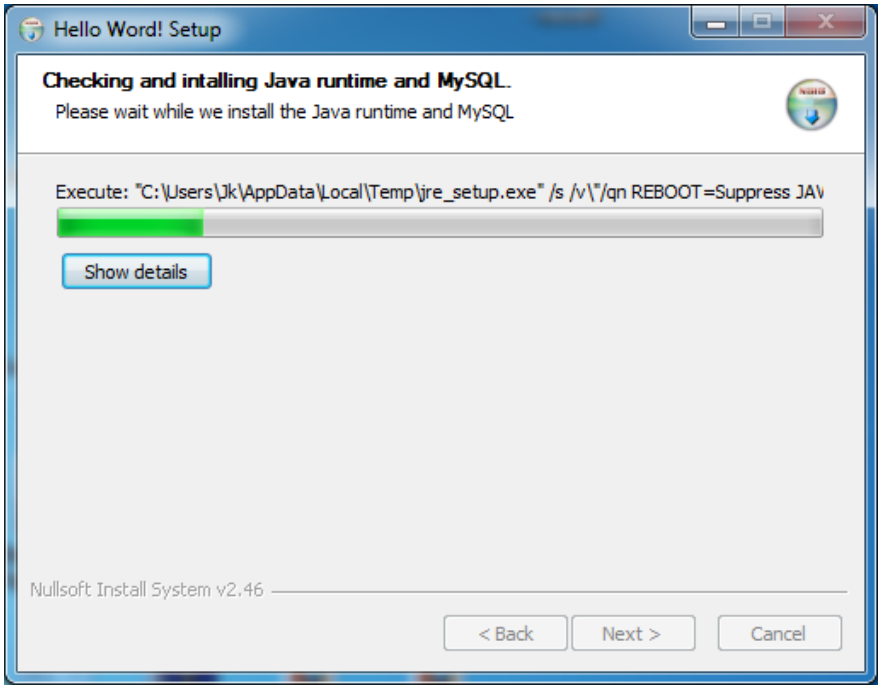


Figura 2- Pantalla 2 instalación JRE

A continuación se comprobará si esta instalada la versión MySQL 5.1, si no es así se abrirá el instalador de MySQL el cual deberemos seguir como se explica en la sección 10.2.2 para instalar y configurar MySQL.

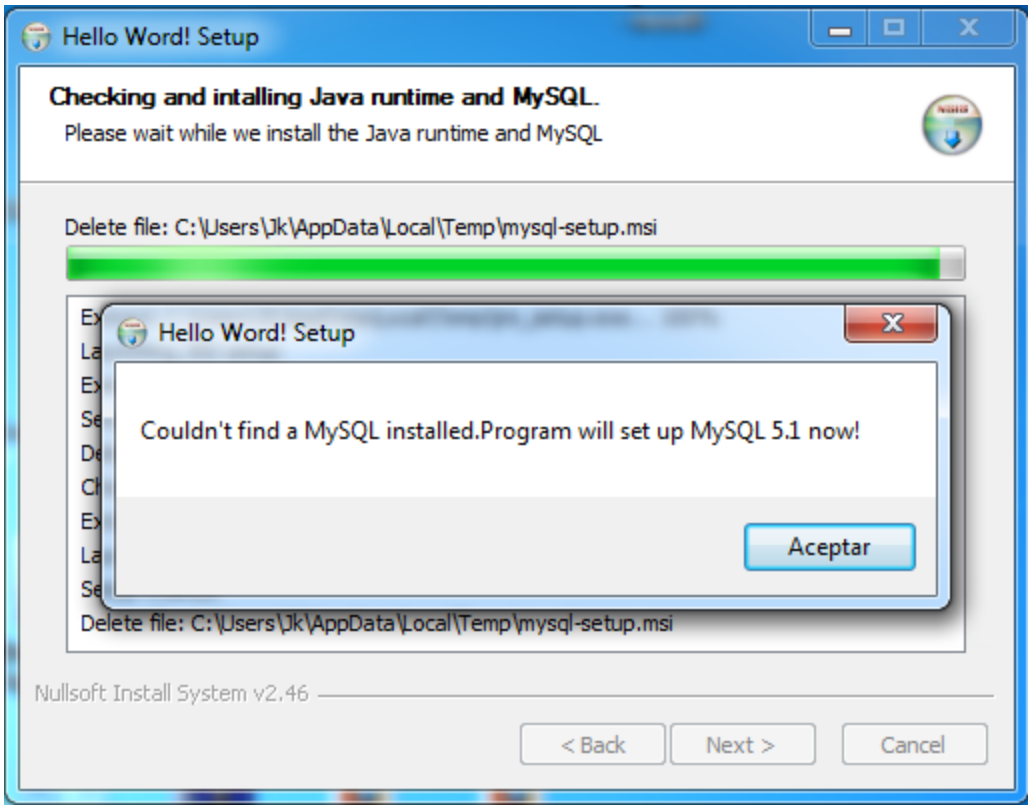


Figura 3- Pantalla comprobación MySQL

Una vez terminado ya nos dejara instalar Hello Word!, y nos pedirá una ubicación para ello. Por ultimo se nos abrirá el directorio y deberemos configurar el fichero configuration.properties como se explica en la sección 10.2.3.

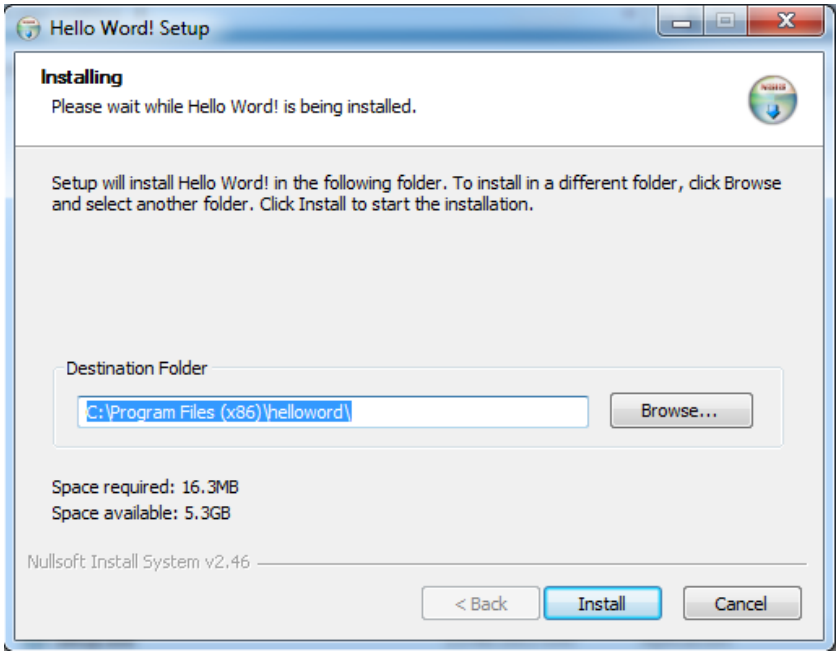


Figura 4- Pantalla instalación Hello Word!

10.2.2 APÉNDICE 2.2. INSTALACIÓN DE MYSQL.

En la instalación de MySQL se nos mostrara la siguiente pantalla:

Figura 5- Pantalla 1 instalación MySQL

Aquí pulsamos “Next”, en la siguiente pantalla seleccionamos “Typical” volviendo a pulsar “Next” y en la siguiente (si no queremos modificar la ruta donde se va a instalar) pulsamos “Install”, esto provoca el comienzo de la instalación del programa, que una vez que termina muestra una pantalla donde tenemos que pulsar en “Finish”.

Una vez hecho esto, vamos a Inicio → Programas → MySQL → MySQL Server 5.1 → MySQL Server Instance Server Wizard (si es que no se ha abierto automáticamente).

Figura 6- Pantalla 2 instalación MySQL

En las sucesivas pantallas que se van mostrando al pulsar “Next” tenemos que seleccionar los siguientes datos (aunque siempre va a depender de nuestras propias necesidades):

- Detailed Configuration
- Server Machine
- Transactional Database Only

- Dejar todo por defecto
- Decision Support (DSS) OLAP
- Marcar la casilla “Enable TCP/IP Networking”, establecer “Port Number” a 3306, marcar la casilla “Add firewall exception for this port” y marcar la casilla “Enable Strict Mode”.
- Seleccionamos la opción “Best Support For Multilingualism” para establecer el encoding de la base de datos a UTF-8.
- Marcar las casillas “Install As Windows Service” e “Include Bin Directory in Windows PATH”, dejando el nombre del servicio por defecto.
- Marcar la casilla “Modify Security Settings”, estableciendo como usuario “root” y como contraseña dejándola en blanco (o la que queramos).
- En este punto es importante recordar el usuario/contraseña porque luego lo necesitaremos para configurar Hello Word!:
    - Usuario: root.
    - Contraseña: La que establezcamos.
- Pulsamos en “Execute” para que comience el proceso de configuración y cuando finalice podemos pulsar en “Finish”.
- Para comprobar que la instalación de MySQL se ha hecho correctamente podemos abrir una consola y teclear “mysql -u root -p”, introducimos la contraseña establecida anteriormente, y el sistema nos tiene que informar con una pantalla parecida a esta:

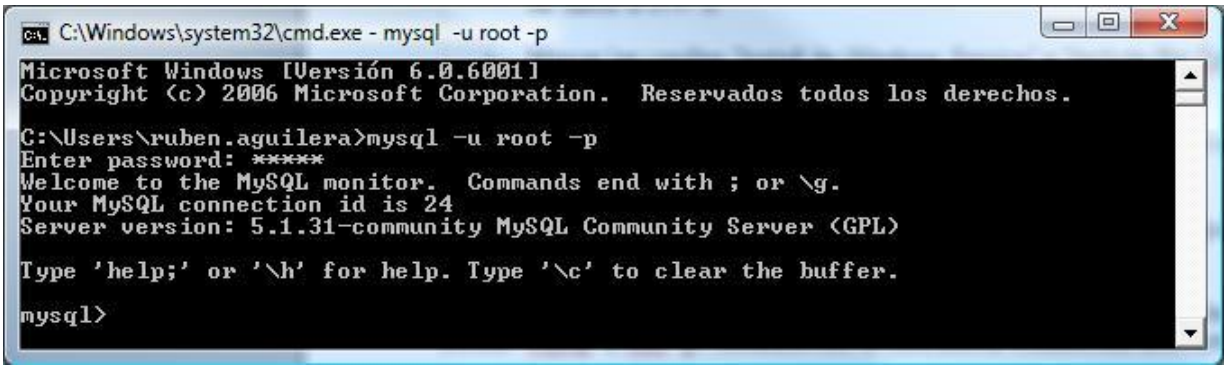


Figura 7- Consola MySQL

Para salir de esta pantalla tecleamos “exit” y ya estaría instalado el motor de la base de datos.

- Posible error en Windows XP.

Es posible que si la instalación la realizamos en un entorno Windows XP al finalizar la instalación se produzca un error: Cannot create Windows service MySQL. Error 0. Estos son los pasos a seguir para solucionarlo:

Desinstalamos la instancia. Para ello, ejecutamos la herramienta "MySQL Server Instance Configuration Wizard", seleccionamos "Remove Instance" y pulsamos "Next".

Accedemos al registro de Windows. Para ello, vamos a Inicio --> Ejecutar y escribimos "regedit". En la pantalla que se muestra accedemos a la ruta HKEY_LOCAL_MACHINE/SYSTEM/CurrentControlSet/Services y eliminamos la entrada Description de "MySQLSer".

Volvemos a repetir los pasos para la instalación.

Sacado de :
http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=InstalacionMySQL5.1Windows

10.2.3 APÉNDICE 2.3 CONFIGURACIÓN DE HELLO WORD!

Por ultimo debemos rellenar los parámetros del fichero configuration.properties encontrado en la carpeta donde instalamos Hello Word!.

Este fichero se debe completar como sigue:

```
#User parameters

profesor.pass=<contraseña que queramos para las funciones del profesor>

admin.pass=<contraseña que queramos para las funciones del administrador>



#Database parameters

Database.url=jdbc:mysql://localhost/

Database.password=<contraseña que pusimos a la base de datos a la hora de instalar y configurar MYSQL>

Database.user=<usuario que pusimos a la base de datos a la hora de instalar y configurar MYSQL, por defecto seria root>
```

Un ejemplo del fichero seria el siguiente:

```
#User parameters

profesor.pass=profe9876

admin.pass=admin1234



#Database parameters

Database.url=jdbc:mysql://localhost/

Database.password=

Database.user=root
```

BIBLIOGRAFÍA

[1] Beck, I.L., McKeown, M.G., y L. Kucan. (2002). *Bringing Words to Life. Robust Vocabulary Instruction*. The Guilford Press.

[2] Blachowicz, C. y Fisher, P.J. (2006). *Teaching Vocabulary in All Classrooms*. Pearson Prentice Hall.

[3] Hiebert, E.H. y Kamil, M.L., eds. (2005). *Teaching and Learning Vocabulary. Bringing Research to Practice*. Lawrence Erlbaum Associates.

[4] Graves, M.F. (2009). *Teaching Individual Words*. Teachers College Press and International Reading Association.

[5] National Institute of Child Health and Human Development (2000). *Report of the National Reading Panel: Teaching children to read: An evidence-based assessment of the scientific research literature on reading and its implications for reading instruction*. Washington.

[6] Stahl, S.A. (2005). *Four Problems with Teaching Word Meanings*. En Hiebert y Kamil, op. cit.

[7] Anderson, R.C., Wilson, P.T. y Fielding, L.G. (1988). *Growth in reading and how children spend their time outside of school*. Reading Research Quarterly, 23, 285-303.

[8] Cunningham, A. E. (2005). Vocabulary Growth through Independent Reading and Reading Aloud to Children. En E.H.Hiebert y M.L. Kamil op. cit.

[9] Schmitt, N. (2000). *Vocabulary in Language Reaching*. Cambridge University Press.

[10] Jessica S. Horst, Kelly L. Parsons y Natasha M. Bryan. *Get the story straight: contextual repetition promotes word learning from storybooks* [en línea]. Available: http://www.frontiersin.org/developmental_psychology_/10.3389/fpsyg.2011.00017/abstract

[11] *Verbal Education Launches Breakthrough Vocabulary Learning System* [en línea]. Available: http://www.marketwire.com/press-release/verbal-education-launches-breakthrough-vocabulary-learning-system-1790497.htm

[12] *Metacognitive Strategy Training for Vocabulary Learning* [en línea]. Available: http://tesl-ej.org/ej26/a5.html

[13] Study looks at how children learn new words [en línea]. Available: http://www.sussex.ac.uk/newsandevents/pressrelease/media/media653.html

[14] Highlighting Is a Waste of Time: The Best and Worst Learning Techniques [en línea]. Available: http://ideas.time.com/2013/01/09/highlighting-is-a-waste-of-time-the-best-and-worst-learning-techniques/#ixzz2UOFXdU1f

[15] John Dunlosky, Katherine A. Rawson, Elizabeth J. Marsh, Mitchell J. Nathan y Daniel T. Willingham. *Improving Students' Learning With Effective Learning Techniques: Promising Directions From Cognitive and Educational Psychology* [en línea]. Available: http://www.psychologicalscience.org/index.php/publications/journals/pspi/learning-techniques.html

[16] Wikipedia. Gamificación [en línea]. Available: http://es.wikipedia.org/wiki/Gamificaci%C3%B3n

[17] Wikipedia. Modelo-Vista-Controlador [en línea]. Available: http://es.wikipedia.org/wiki/Modelo_Vista_Controlador

[18] Wikipedia. Java [en línea]. Available: http://es.wikipedia.org/wiki/Java_%28lenguaje_de_programaci%C3%B3n%29

[19] Wikipedia. MySQL [en línea]. Available: http://es.wikipedia.org/wiki/MySQL

[20] Hibernate [en línea] Available: http://www.hibernate.org/

[21] Patrones de diseño. *Guía de construcción de software en java con patrones de diseño. escuela universitaria de ingeniería técnica en informática de Oviedo* [en línea]. Available: http://di002.edv.uniovi.es/~cueva/asignaturas/PFCOviedo/PFCpatronesJava.pdf

[22] Caba Vargas (2000). *Enseñanza individualizada, evaluada por mastery, asistida por computadora* [en línea]. Available: http://hdl.handle.net/123456789/1143